KNOWLEDGE ENGINEERING APPLICATIONS IN AUTOMATED STRUCTURAL DESIGN

By

NAGARAJAN SHANKAR

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1991

To my parents

and

Ravi & Hari

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Prabhat Hajela for his encouragement and support throughout the research. I would also like to thank Drs. Nevill, Navathe, Sun, and Zimmerman for consenting to serve on the thesis committee.

I am extremely thankful to my parents for their blessing and enduring support during my graduate studies. I also thank all my friends who made my stay in Gainesville enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

KNOWLEDGE ENGINEERING APPLICATIONS IN AUTOMATED STRUCTURAL DESIGN

By

Nagarajan Shankar

May 1991

Chairperson: Dr. Prabhat Hajela
Major Department: Aerospace Engineering, Mechanics and Engineering Science

The present work focuses on the development of AI based design systems. The work proposes a general methodology for integrating AI methods with algorithmic procedures for the development of preliminary configurations of load bearing truss and frame structures. The proposed framework is an adaptation of a recently proposed model for generic problem solving systems. The methodology is characterized by the use of algorithmic processing to enumerate the design space, and utilizing this information in a design process dominated by heuristics.

The design process deals with the synthesis of preliminary structural configurations for a given problem specification. The approach is based upon the systematic decomposition of the design space, and repeated application of design heuristics to incrementally generate an optimal structural topology to account for the

applied loads. The principal shortcomings of a heuristic decomposition approach are underscored, and more rational quasi-procedural methods are proposed.

The research sheds light on two distinct methods of decomposition for large structural design problems. The first method which has its basis in global sensitivity analysis, proposes an approach to decompose the design space into subproblems for which solutions can be readily obtained, and to integrate these partial solutions into the system solution by consideration of first order sensitivities of subproblem interactions. The second approach employs the use of dynamic programming, wherein the size of the solution space is managed by a decomposition of the design problem into many stages.The use of this strategy generates a number of possible configurations, which are then subjected to a genetic algorithm based stochastic search to yield a near-optimal structure. Several design examples are presented to demonstrate the validity of the design methodology and the decomposition methods.

CHAPTER 1
INTRODUCTION

Background

Optimal structural design for minimum weight is a multistage process, initiated with the definition of a structural topology to transmit applied loads to specified support points. Topology generation forms the initial and perhaps a crucial step in the design process. In most problems, there is no unique topology for a given load-support specification. Once an acceptable topology has been generated, the members of the structure are resized to satisfy structural and dynamic response requirements using formal optimization techniques. The final outcome of this procedure is a minimal weight structure, the characteristics of which are strongly linked to the original topology definition.

Generic CAD systems available today are generally limited to problem solving tasks in the algorithmic or deterministic domain. However, creative aspects in design are often ill structured, and require significant intervention on the part of the designer. Therefore, attempts have been made in recent years to incorporate decision making capabilities in automated design procedures; the topology definition problem described above is naturally amenable to this approach. Such decision

making characteristics warrant the use of artificial intelligence methods with unique abstraction capabilities. Artificial intelligence methods, such as search techniques and expert systems, allow us to manage problem decomposition and knowledge representation in a very efficient manner. Additionally, expert systems help in the dissemination of design procedures to a large user community. Use is made of the knowledge and experience of the human designer in this process. However, there exist two major drawbacks of such decision support systems as far as optimal structural design is concerned. The first pertains to the rather limited computational power of such systems which renders them inefficient for a meaningful structural design problem. The low computational efficiency is usually attributed to the language selected for the development of such systems. The second problem involves its limited interfacing capabilities with algorithmic computing. Therefore, most of the expert systems developed for structural design do not extend into the analysis and optimization domains. Hence, the development of systems, which realize the advantages of integrating heuristic and algorithmic computations, has to be explored.

The common feature to all the knowledge based systems was interaction with the user. The user responds to a series of queries posed by the system, and the system uses this information in conjunction with the knowledge built into the system to arrive at pertinent conclusions. Therefore, such structural design systems do not comply with the accepted definition of automation. Hence, attempts have to be made to deemphasize interaction, to achieve a larger degree of automation.

Problem decomposition plays a major role in the design process. The basic

theme of decomposition is to divide and conquer. The design process involves successive refinement in which the problem is decomposed into simpler, preferably single goal problems. Ultimately, the solutions to all the single goal problems are integrated in a rational manner. Decomposition methods have been employed in the design of complex structures, but the dependence of design on such methods has not been completely studied. A study of problem dependence is of paramount importance in establishing the right approach for problem decomposition.

The aim of the present work was to study various facets of heuristic methods in automated structural design. First, an attempt was made to provide a formal architecture for the proposed design system. Second, this architecture, which provided for automation and involved both heuristics and algorithmic computing, was implemented to design an optimal structure for given load-support specifications. Third, decomposition methods with a basis in dynamic programming and sensitivity analysis were studied, with the goal of enhancing the strategies of problem decomposition. Also included in the decomposition studies was an examination of genetic algorithms as a viable optimization approach for large scale structural system synthesis. Emphasis was placed on heuristic reasoning which was accomplished using the CLIPS rule based expert system environment.

Relevant Literature

Numerous knowledge based systems have been developed in the area of

design of engineering systems. A general observation about most such systems is that there is limited emphasis on attempting to optimize the final design.

Ohsuga [1] discussed the conceptual design of a CAD system, which emphasized the importance of representing and processing knowledge to enhance modeling and manipulation. The design system does not address the optimization of the final design. Airchinnigh [2] addressed the problem of interfacing knowledge systems and conventional CAD software. He discussed the need for development of a universal programming language for both CAD software and knowledge representation software.

Rehak et al. [3] discuss a possible architecture for structural building design, based on a distributed network of knowledge based processing components. However, the architecture does not address the automation of the iterative process. Shah and Pandit [4] discuss the synthesis of forms for machine structures. A taxonomy of primitives serves as the component database, and an expert system synthesizes forms using these primitives. This is one of the first reported works in the area of preliminary design synthesis. Nevill et al. [5] reported the importance of problem abstraction in preliminary synthesis. This work involved the implementation of a design methodology for 2-d mechanical and structural systems. However, there is no reference to algorithmic computing in the reported approach. Brown [6] studied the treatment of subproblems in preliminary design. This work discussed the dependence of preliminary synthesis on design abstraction. Brown and Chandrashekaran [7] report the development of an expert system for the design of

mechanical components. The problem of design refinement is treated using heuristics and a hierarchical representation; however, optimization of the final design abstraction is not addressed. The paper sheds light on the advantages of hierarchical representation of the design process. Davies and Darbyshire [8] present an expert system for process planning to decide metal removal in a manufacturing system. In case of process planning, where algorithmic methods of solution tend to be inadequate, expert systems are proposed as the viable alternative. Rooney and Smith [9] present an investigation of a design methodology with an emphasis on the usage of AI methods. This is one of the earliest attempts at AI based design methods but does not include the concept of optimization as an end module in the procedure. The key feature is the design knowledge being reintroduced as an integral part of a data management facility. Dixon and Simmons [10] reported one of the first attempts to introduce an expert system as a viable tool in the design of mechanical systems. The work also emphasized the role of conceptual design, validation, and evaluation of the derived design, to be represented as information modules for the expert system. Liu [11] suggested the importance of integrating the decision making capabilities of knowledge based systems into mainstream design methods. The advent of supercomputers in the design workplace can be viewed as complimenting the strength of expert systems, and both systems in tandem can be effectively employed in the engineering decision process. The approach suggested in the paper is robust and can be easily implemented. The importance of supercomputers in uncertain domains like the design synthesis problem is underscored. Arora and

Baenzinger [12] report the implementation of AI methods in tandem with optimization methods. Heuristics are used to analyze the results after each iteration and optimization process is enhanced. Bennett and Englemore [13] report an attempt in the development of expert system for structural analysis, SACON. This was the first reported work in the area of structural analysis, and can be categorized as a consultative system. Rogers and Barthelemy [14] discuss an expert system implemented to aid users of the Automated Design Synthesis (ADS) optimization program. Sriram et al. [15] describe an implementation of design methodology integrating expert systems in structural design for civil engineering. The analysis system does not emphasize the use of optimization to derive the final design.

Expert systems have also been successfully applied in manufacturing. Most of these applications have been in the form of diagnostic tools and performance monitors for machinery. Lusth et al. [16] present a knowledge based system for arc welding. This work reported the experiences in development of a system to aid in the welding of thin metal plates. Liu [17] reports a knowledge based system for finite element modeling for strip drawing applications. The paper highlights the use of knowledge based systems in other areas of mechanical engineering. Shodhan and Talavage [18] present an AI based approach to manufacturing system design. The simulation of a manufacturing process including selection of design, manufacturing process, and inspection method is aided by the incorporation of a knowledge based system. Phillips et al. [19] report research efforts in integrating AI methods with existing CAD and CAM technologies. A knowledge based system is used to convert

the design for manufacturing. Murthy et al. [20] discuss a methodology for the organization of design data and relevant information for use with knowledge based design methods. A graph based data representation scheme has been reported for use with primitives based design. The scheme can be translated into a relational database system for use with quasi-procedural design methods. In general, most of the aforementioned knowledge based systems are of a consultative type, which impedes the total automation of the design task.

A careful analysis of the relevant literature in knowledge based design identifies two important drawbacks. The first is a lack of formalism in the development of a design architecture for structural design, although most of the knowledge based design systems contain similar computational modules. Tong [21] identified the need for formalism in knowledge based design and proposed a general framework for knowledge based problem solving systems, recognizing the type of information involved. The second is that a formal attempt to derive an automated design procedure, including all formal stages in the design process and a conceptual design module in the form of a knowledge based system, has not been successfully accomplished. One possible explanation for the latter could be that the system tends to get very complex, considering that a tedious finite element procedure has to be incorporated for design analysis. Issues such as problem decomposition and alternative optimization methods have to be addressed in the context of available efficient AI methods. Such efforts would probably assist in the complex process of structural design.

Prototype Design System

Figure 1-1 indicates a conventional design methodology for the design of structural systems. The different stages indicated in the design constitute the standard methodology adopted in a typical design procedure. The first phase relates to the generation of an initial design, which is typically either an existing design or one which evolves through heuristics derived from prior experience. The design, which is primitive at this stage, is modified in a detailed design process to meet all the requirements.

The next phase actions relate to the analysis and subsequent changes in the design from the previous step. The analysis of the design is carried out to evaluate the design constraints and check for their satisfaction. If all the constraints are satisfied, then the design is considered feasible and can be promoted to the optimization phase. If the constraints are not satisfied, then the design is iteratively modified through the use of heuristics or the application of formal optimization methods. In case a design is modified using heuristics to achieve a satisfactory design, a procedural optimization derives the final design. This is iterative also, and the final outcome is an optimal design. The methods adopted are usually gradient based, although the role of new algorithms based on evolution theory are presently under examination.

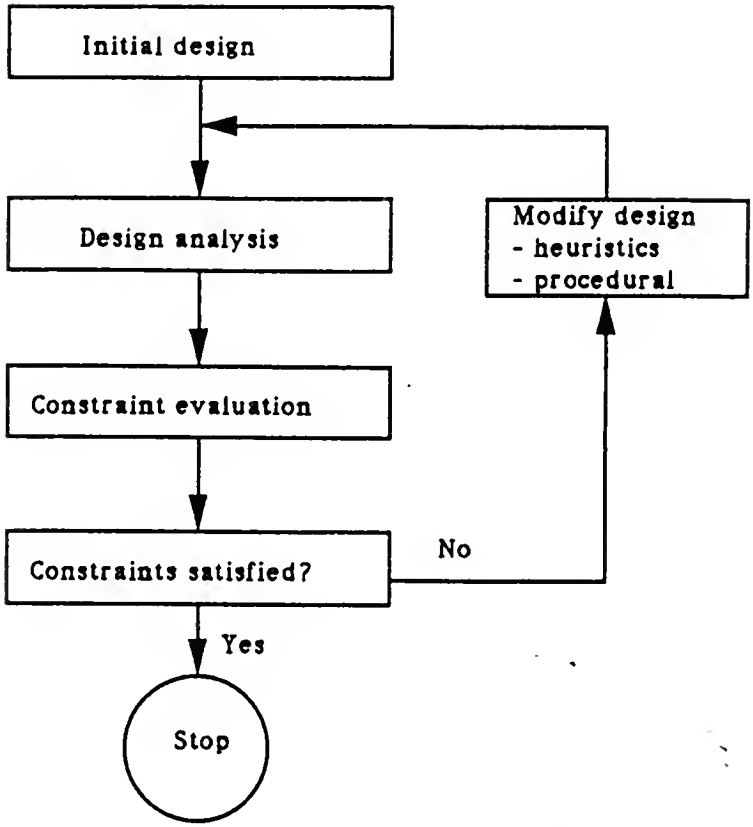


Figure 1-1. Conventional design methodology.

Expert Systems

Expert systems are computer programs which simulate the actions and decisions of a proven expert in a given domain. Architecture of a typical expert system and the interactions between the different modules are shown in figure 1-2. The main ingredients are a data storage capability, a knowledge representation scheme, a mechanism to operate on the knowledge and arrive at decisions, and last but not the least, a user-friendly interface.

The knowledge representation module in figure 1-2 is a representation scheme for all the domain specific information in an usable form, depending on the application involved. There are many ways of representation, with the frequently used method being in the form of IF... THEN type rules. All the domain relevant heuristics are stored as rules and are referred to as the 'knowledge base' of the expert system.

The data storage module for the expert system contains all the relevant information pertaining to the existing conditions. This is also referred to as the database. Information retrieval and storage is performed from this database. External databases can be utilized for large scale expert systems.

Inference module is an integral part of the expert system. This is a mechanism which operates on the knowledge base, utilizing information relating to the existing conditions in the problem domain, to provide decisions that may effect those conditions. There are two recognized methods in inferencing, namely forward

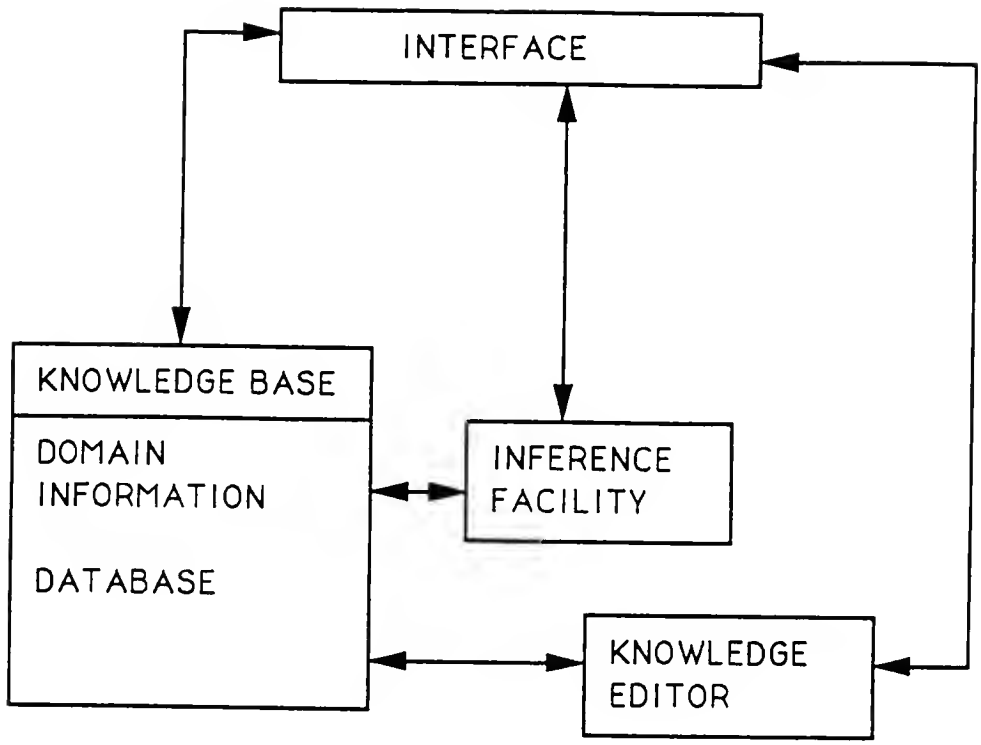


Figure 1-2. Schematic of a typical expert system.

and backward chaining. The forward inferencing method is best suited for problems such as design, where there are numerous outcomes for a given set of initial conditions. Backward inferencing is preferred when a well defined set of conditions is known, and events that led to those conditions are known to occur.

The user interface is an added component to the expert system, to make it amenable to the user. Extensive graphics are usually incorporated in support of such a user interface.

Overview

A modification of the conventional structural design approach, to incorporate the advantages of established AI tools, is the principal focus in chapter 2. An AI based design framework is outlined which recognizes the various levels at which information must be organized to facilitate an effective integration of heuristics and procedural methods. All the tools and procedures are classified depending on the type of information being processed, and their interactions are defined. Chapter 3 details the implementation of the proposed AI framework. Preliminary examples which illustrate the effectiveness of such an implementation are presented. A decomposition method based on the recently developed global sensitivity analysis is the focus in chapter 4. Details of the development and implementation of the method, including illustrative design problems are provided in this chapter. A stagewise (SG) decomposition approach which is thematically similar to dynamic

programming (DP) is presented in chapter 5. Also described in this chapter is a genetic search (GA) approach to optimization, utilized for member resizing.

An altogether different scheme for structural topology synthesis, which is a consequence of the SG/GA approach of chapter 5, is the focus in chapter 6. In this chapter topology generation and optimization are considered from the GA point of view, with support stemming from heuristics directed SG. The chapter describes the principal issues in such an approach. Finally, concluding remarks and some directions for further research are presented in chapter 7.

CHAPTER 2
DEVELOPMENT OF DESIGN METHODOLOGY

The development of a framework for a design system, which could act as a template for future implementation of knowledge based structural design systems, was the first objective in the present work. This framework was an extension of the general approach provided by Tong [21] to encompass the structural design domain. The underlying approach is quasi-procedural, wherein heuristics suggest a design which is then verified by an algorithmic process.

Generic Problem Solving Systems

The process of automated conceptual design proceeds with a synergistic application of design heuristics and domain analysis programs, resulting in a tightly coupled quasi-procedural approach. The organization of a problem solving system to achieve this task determined the overall efficiency of the process. Separation of various components of such a system was important, as it retained a level of modularity in the system which allowed for component updates. At the very outset, it was important to recognize three distinct levels at which the automated design process was organized--the knowledge, function, and program levels. A typical

arrangement of the three levels in relation to each other is shown in figure 2-1. The figure also portrays typical component modules for the different levels.

Knowledge Level. A detailed description of the design domain is relegated to this level. This included all pertinent specifications which define the acceptability of a solution. All necessary and applicable requirements for analysis are also made available at this level. Furthermore, the types of design applications envisaged for such systems would very seldom generate designs that bear no resemblance to their predecessors. A possibility exists, therefore, to develop at this level a general taxonomy of problem types, of the most typically encountered design constraints, and of possible solution strategies. The domains that can be considered in this exercise include geometric modeling, structural analysis, and optimization methodology.

In creating a taxonomy of designs based on problem specifications, one is essentially identifying abstract features that result from an imposition of such specifications. Structural design requirements may be classified on the basis of strength, stiffness, elastic stability, degree of redundancy, types of support conditions, dynamic behavior, and a requirement of least weight or least complexity in manufacturability. Clearly, each of these requirements has an influence on the design that distinguishes it from designs dominated by other requirements. As an example, a structure that is governed by structural stability requirements will be dominated by elements that can withstand compressive loads, and further, such elements will typically have appropriate aspect ratio and stiffness properties.

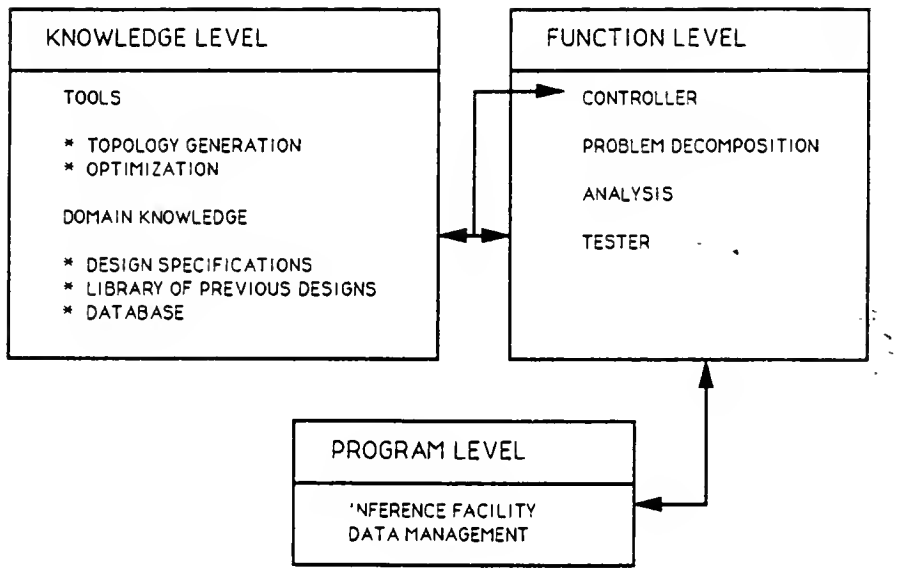


Figure 2-1. A general framework for design.

However, in as far as possible, it is advantageous to relate one abstract feature with one problem requirement. Failing this, the classification must clearly indicate the relative contribution of a requirement to a salient characteristic.

Clearly, the information available at the knowledge level determines the class of problems for which solutions can be obtained. The coupled algorithmic and heuristic process can be computationally demanding in some situations. It is in such cases that a taxonomy of designs based on problem requirements is particularly useful. It is conceivable to abstract partial designs in a primitive form, where such primitives adequately model the displacement field of more refined models, but may have considerably fewer structural elements. These primitive forms may be retained in the analysis to a point where it becomes necessary to introduce greater refinement in order to meet a specific class of design requirements. One can think of a macroelement that has specific stiffness characteristics to satisfy displacement requirements. Such a macroelement can then be defined in terms of component elements with the same aggregate stiffness, and which account for strength requirements.

All the heuristics that aid in the synthesis of topology are maintained at this level. This includes the heuristics employed for design decomposition and subproblem integration as well as for optimization purposes. Constraint requirements that are to be satisfied, if any, by any chosen topology, are retained here. A global database, which stores domain information, taxonomy of topologies, allowable primitives and design requirements, is also provided at this level.

Function Level. Perhaps the most significant feature of this level of organization is a controller which directs the flow of the conceptual design process. The controller is essentially an embodiment of the problem solving strategies that may be invoked for the problem at hand. The design specifications handed down from the knowledge level are attempted to be satisfied at the function level. Although designs can be generated by considering all requirements simultaneously, this methodology is not considered appropriate for the task at hand. Instead, a more natural process of refinement in steps is adopted, wherein the problem is decomposed into smaller, more tractable, and preferably, single goal problems. The underlying principle in such a refinement is that the solution space is more likely to be unique in the presence of a higher degree of specification detail.

In a problem where the design philosophy is one of sequential refinement to satisfy an ordered set of goals, there is a need for evaluating the proposed partial concepts for acceptance. Such testing operators are made available at the function level and simply interact with the available information base at the knowledge level to recover the pertinent information. The failure of a proposed concept to meet the acceptability test is generally followed by alternative design moves. Such moves are facilitated by the initial decomposition of the problem by design specifications; this allows for construction of tree-like deduction paths.

Finally, the controller must have the option of modifying the design rules, particularly if it assists in realizing the design specifications. The acceptability tests can themselves be relaxed to admit marginally infeasible designs. Yet another option

available at this stage is to extend the design by adding features which enable it to pass the acceptability requirements. It is important to understand that for any given task, there may be several transfers of control to the program and knowledge level as deemed appropriate by the controller at the function level. This transfer of control was repetitively invoked for each of the component tasks.

Pertinent tools available at this level included domain descriptors, analysis modules, dynamic programming and genetic algorithms for design decomposition and optimization, respectively, and other modules for nonlinear optimization.

It is important to emphasize the interactions that exist between this level and the other two levels. Since, the function level constitutes all the required evaluation and problem description tools, it has an implicit dependence on the program level, which in turn couples to the knowledge level.

Program Level. At this level, no problem solving knowledge is explicitly available. However, the implementation of all design steps, on the basis of information received from other levels, is carried out at this level. In addition, tasks of data management and programming procedures are assigned to this level. The database management capabilities of such systems are of particular importance: significant amounts of data are generated and must be managed for a design system to work efficiently. This is even more crucial as large amounts of algorithmically generated numerical data must be stored and post-processed to be used effectively in the iterative process.

The inference facility is perhaps the most significant feature at this level. The inference approach in the current implementation was forward chaining, which is a logical choice considering the fact that the design process has numerous solutions. In structural design problems that constitute the focus of the present work, the inference mechanism available in the rule-based C Language Integrated Production System (CLIPS) [22] is used. This utility can be invoked from within a Fortran program, making available a convenient link between algorithmic and heuristic processing of information.

Since the design process is inherently complex and tedious, a good I/O capability is an essential ingredient. In the present development, emphasis is placed on design automation and hence a very minimal I/O interaction is permitted. The key I/O capability is a graphical display of the generated topology, and a topology plot file was created during the design process. Once the design procedure is completed, the plot file can be displayed to view the topology.

In keeping with the proposed framework of developing generic problem solving systems, the various tools necessary for the task were assembled into each of the three levels of organization as shown in figure 2-2. Appendix A presents a detailed representation of each module within a particular level of organization.

Problem Definition and Primitives

Figure 2-3 shows a representative design problem considered for structural

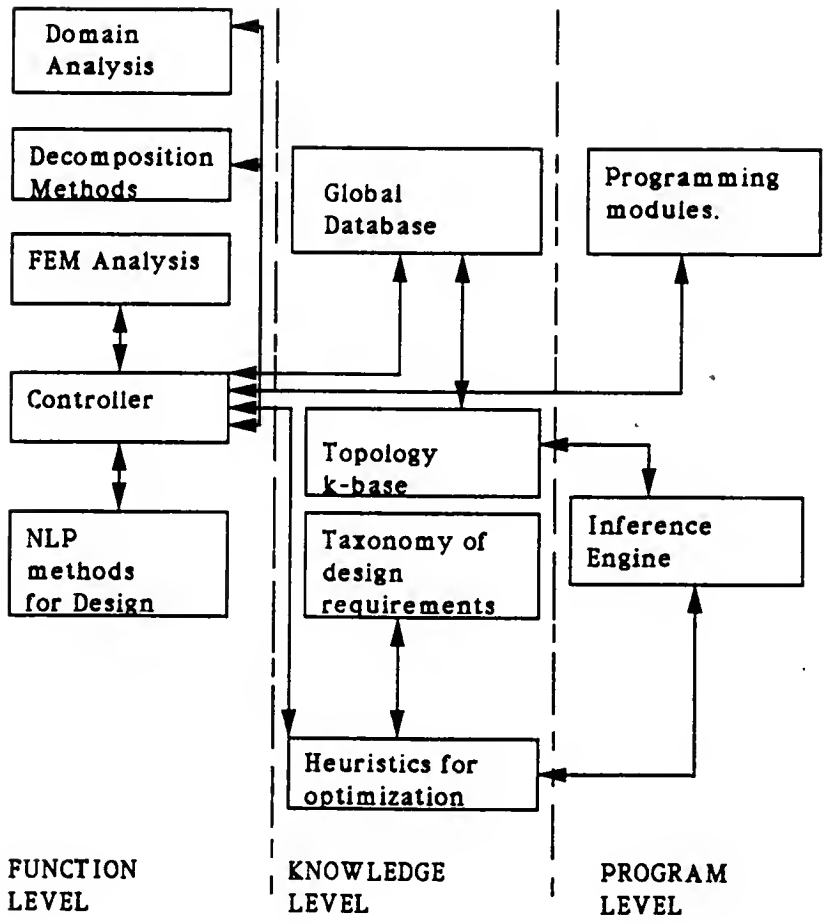


Figure 2-2. Organization of the three levels.

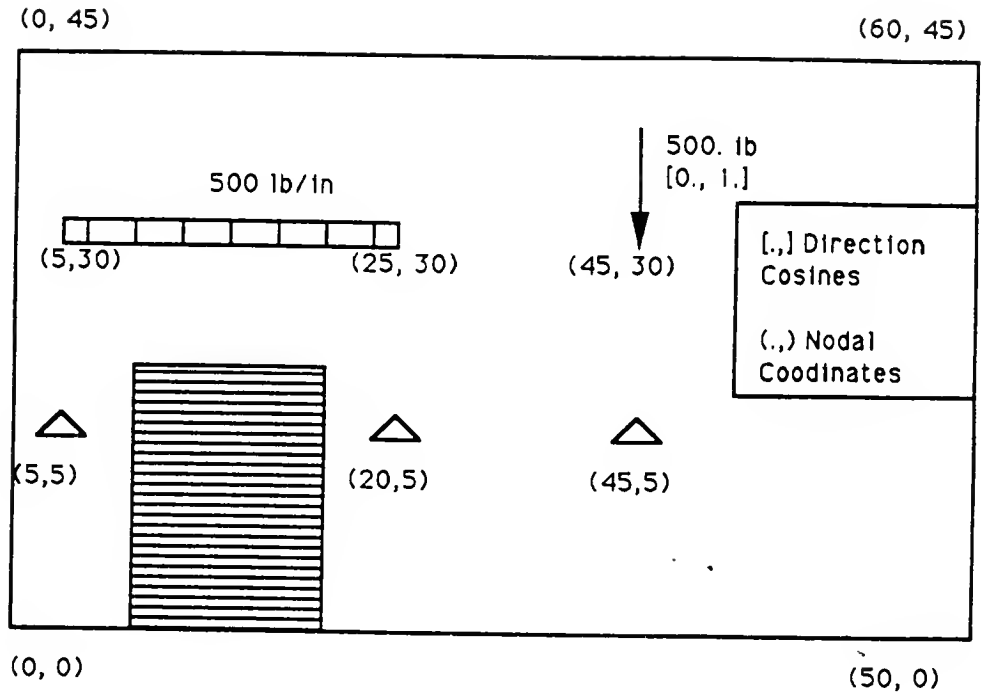


Figure 2-3. Typical design problem.

synthesis. The domain consists of a set of concentrated and distributed loads, moments, possible support points, and regions where structural members are not permitted. The task involved the determination of a least weight structural topology to transfer the given loads to the prescribed support points, and avoiding the forbidden zones to whatever extent possible. The design was performed in conjunction with analysis methods to evaluate the relative merits of each topology. The topologies were designed for constraints dictated by displacement or stiffness requirements, strength requirements, and for dynamic considerations as represented in frequency constraints.

Such a problem usually incorporates a set of prescribed primitives, from which the structure is evolved. Table 2-1 lists the various primitives that were considered for loads, supports, materials, etc. A database of all the primitives with corresponding properties was maintained, and from which relevant information was sought during the design process.

Optimal Structural Synthesis

The framework described in the preceding sections was implemented in an automated design system for the generation of near-optimal, two dimensional structural configurations. The problem required the development of both the structural form and member cross-section dimensioning, to carry a prescribed set of concentrated and distributed loads and moments. The least weight structural weight

Table 2-1. Table of primitives.

Primitives	Attributes
Element	Beam, Truss
Load	Point Loads, Moments, Distributed Loads
Support	Hinge, Clamp
Material	Steel, Aluminum, Aluminum Alloy
Cross Sections	I-section, C-section, Tube, Rod

was sought with constraints on allowable displacements, fundamental frequency, and structural strength. The normalized constraints for the displacements, natural frequency and stress response can be expressed as follows:

$$\frac{d_i}{d_{all}} - 1 \leq 0 \qquad (2\text{-}1)$$

$$1 - \frac{\omega_i}{\omega_{all}} \leq 0 \qquad (2\text{-}2)$$

$$\frac{\sigma_i}{\sigma_{all}} - 1 \leq 0 \qquad (2\text{-}3)$$

where the subscript 'all' denotes allowable value of the response quantity. In the present work, these limits were set to the following values:

$$d_{all} = 10^{-4} \text{ in}$$

$$\omega_{all} = 12.5 \text{ Hz}$$

$$\sigma_{all} = 250. \text{ psi}$$

In addition, side constraints were established for all design variables (x_i) as follows:

$$0.01'' \leq x_i \leq 5.0'' \qquad (2\text{-}4)$$

At the function level, the topology analysis module was responsible for developing a table of pertinent features for each of the loads and supports. These

features include load-load distances, load-support distances, their angular orientations, nature of the loads, type of supports, and information regarding location of forbidden zones. Two other modules at the function level were programs for finite element analysis, and a nonlinear programming optimization program. The latter is available to perform procedural/ heuristic optimization on a given geometry. Also present at this level was a controller which directed the flow of the synthesis process. Two distinct procedures were adopted in the present work. In the first phase, an arbitrary decomposition of the problem was prescribed to the controller, wherein the structural design proceeded in ordered steps of satisfying the displacement, frequency, and stress constraints. More rational decomposition strategies based on dynamic programming and global sensitivity methods constituted the second phase of the work and are described in chapters 4, 5 and 6.

Data Management

The process of structural design described in the preceding section cannot be successfully implemented without reference to an extensive and detailed description of the features of the design space. Such large quantities of information required an efficient data handling capability. In the present framework, this data management facility was provided at the knowledge level. A global database is central to the knowledge level organization. Figure 2-4 represents the data organization in the present framework. Both domain and design specifications were resident in this

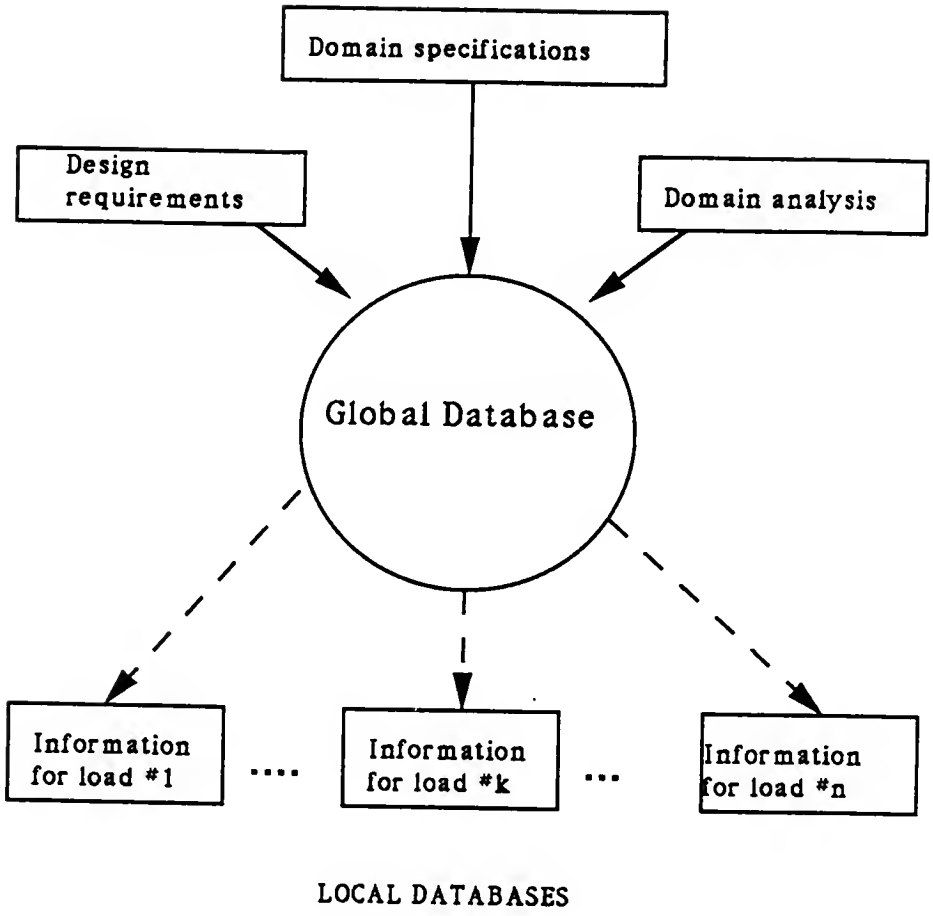


Figure 2-4. Data management facility.

database, as were the results obtained from any enumeration of the design space during the process of sequential refinement. Local databases were created from this global database for use in both algorithmic and heuristic processing in the knowledge and program levels. The creation of these local databases was closely linked to passing of control from one level to another. A typical local database consists of information pertaining to a particular load, such as load-support distance, type of support, forbidden zone information, etc. The advantage of creating a global database is that the information can be utilized any time later to either alter the existing design or synthesize a new design.

Two levels of data management were implemented in the current system. At the core level was a global database that records information for long term usage. In the current implementation, this was achieved by simply writing all the information pertaining to a load such as the distances, orientations, supports, etc., for all the loads into a data file. Pertinent information was retrieved from this data file when necessary. Problem and subproblem related databases were extracted from the main data file and were local to the knowledge level. Such an approach provided a convenient blackboard for constraint posting and propagation as the design was taken through a process of incremental refinement.

CHAPTER 3
IMPLEMENTATION OF DESIGN METHODOLOGY

This chapter describes the implementation of a design system conforming to the general framework presented in the previous chapter. All the relevant implementation details are discussed with reference to illustrative examples. The system was implemented on a VAX 3100 platform, utilizing the VMS Command Language feature wherever possible for automation. Issues ranging from representation of heuristics, to development and integration of procedural and knowledge base [CLIPS] modules, are described in the following sections.

Knowledge Representation in CLIPS

All the design heuristics were represented in the form of rules, written in a format required by the inference environment of CLIPS. A general rule syntax in CLIPS is given as follows:

```
(defrule <name> ["comment"]
        (<first pattern>)
        [(      ..      )
                ..                      ;LHS
                ..
```

```
        (<nth pattern>)]
=>
        (<first action>)
        [(      ..      )           ;RHS
         (      ..      )
        (<nth action>)])
```

The LHS consists of one or more patterns which form the condition(s) to be satisfied, before the rule can be fired. An implicit 'and' is present in the LHS if more than one pattern is present. The RHS of the rule is a list of one or more action(s) to be performed as a result of the firing of the rule. The rule base also included a few meta-rules; these were also treated as rules in the inference environment, each of which is a superset of two or more rules. The advantage of using meta-rules is the efficiency which results from concatenating task specific rules. Furthermore, a better control over the evaluation of rules is possible.

Both an interactive mode of execution and one in which the expert system is embedded in a set of algorithmic and procedural programs, are available in the present implementation. Initially, all illustrative examples were run interactively. This provides the opportunity to view the execution process, editing the knowledge base, and the ability to maintain a record of the status of the system. A typical usage involves creating a rules and facts file, which are used during CLIPS execution.

The other mode of execution is the use of CLIPS as an embedded expert system. In this mode, CLIPS is executed from a FORTRAN program, preceded by

the loading of the rules file. All later implementations involved embedded CLIPS execution. A basic feature in the CLIPS environment is that rules and data are two separate entities. The inference engine uses a forward chaining algorithm to apply the knowledge (rules) to the data.

In a basic execution cycle, the knowledge base is examined to see if conditions of any rule are satisfied. This is done by simple pattern matching of the data with the conditions. If the fields of the data which are asserted as variables, match the conditions of a rule, then that rule is activated for firing. In case of more than one matched rule, all such rules are activated and pushed onto a stack. The most recently activated rule has the lowest priority and is put at the bottom of the stack. The top rule is selected and executed. As a result, new rules may be activated and some previously activated rules may be deactivated. This cycle is recursively applied until no further pattern matches are obtained.

Knowledge Base for Topology Generation

The definition of the design domain was placed into the knowledge level. In particular, this included specification of the magnitude, type, and orientation of the loads with respect to the supports in the chosen reference axes system. The type and location of support points were also part of such a database. This extended database was used in conjunction with the stored knowledge base to invoke the necessary steps in generation of the structural topology.

The underlying philosophy in the present sequential procedure was to stabilize one load at a time. The pertinent load information was retrieved from the database. If the applied load happened to be a moment, one design decision was immediately obvious. The element that transferred this load to a support must allow for rotational degrees of freedom at its end nodes. Hence, a beam element was selected to transfer this load. To stabilize this load and to maintain a minimum weight, the nearest support was examined. If this support was of clamp type, then a beam element of nominal cross sectional dimensions was chosen. In case the support was not a clamp, then the next nearest support was considered. If this support was a clamp, then a beam element was chosen between this support and the load. If not, the load was connected to both the supports using two beam elements. A further level of refinement is possible by checking to see if the third nearest support is a clamp, and if so assess the weight penalty of connecting to this support against the use of two beam elements to the two nearest supports (a frame configuration).

This strategy was also used in the event that at a given point, both applied forces and moments were present. For the situation where the loads are the only applied forces with nonzero x and/or y components, the orientations of the loads stored in the database were used. The angle between the force resultant and the hypothetical element connecting the point of load application and support, was computed. If this angle was less than 20 deg, then the load was considered to have a large axial component, and an axial rod element was preferred.

The other possible case was one in which a large transverse component of the

load existed in addition to an axial load. The orientation of the load resultant with respect to the next nearest support was considered. In the event that the load could be considered to be largely axial for such an element, an axial rod element was introduced between the load and this support. In the event that this condition was also violated, the first two supports were considered. If the nearest support was a clamp, then a beam element was chosen between the load and support. If not, then a beam element was chosen between the load and the second support, provided it was a clamp. Otherwise, two beam elements were chosen between the load and the two supports. Multiple loads acting at a point were simply resolved into the x and y components, and handled as above. Likewise, distributed loads were replaced by equivalent end loads and moments, which were computed by assuming the distributed load to act on a beam segment, and these loads and moments were connected to supports as explained in the foregoing discussion.

Once a load was stabilized by the process described above, the point of application of the load became available as a support point. This is physically an elastic support but may be modeled as a simple support point, for subsequent generation of structural topology. In case of truss structures, stability was verified based on the relationship between the number of members, nodes and degrees of freedom. The controller used the above heuristics until all the loads in the input had been accounted for and were stabilized.

CLIPS Integration

As described in an earlier section, the input to the design process included a specification of loads, type and location of supports, and specification by coordinates of any forbidden zones in the design domain. Using this input, a FORTRAN program was used to extract all relevant data about the design domain such as distances between loads, between loads and supports, any possible non-permissible members, and the orientation of loads with respect to available supports. One of the major tasks for this module included the creation of a facts file. This file contained the description about each load, as shown below.

(load#, load location, x component, y component, moment, support#, type of support, support location, distance, angle, (next nearest support information). . . .)

A CLIPS batch file was created which uses the facts and the rules to generate a topology. The topology generated was stored in a data file in a two dimensional array as shown below.

member	load#	support/load#	type
1	1	3	b
2	1	2	-
.	.	.	.

The array shown above contains the connectivity information of the structure. The symbol in the column 'type' denotes the type of member between the specified load#

and the support/load#. The allowable symbols are 't' for truss, and 'b' for beam elements. The numbers in the array indicate the specified member#, load#, or support#.

Once the topology was generated so as to stabilize all loads, the completed structure was available for finite element analysis to check satisfaction of the design constraints and possible modification of topology using heuristics, as described in an earlier section. In each of these incremental modifications to obtain constraint satisfaction, the controller used the output of the CLIPS based topology generation routine, and created an input runstream for the EAL (Engineering Analysis Language) [23] finite element analysis program. Upon execution of this program, the necessary constraint and objective function information was read and returned in the proper format for it to be read as a CLIPS input file. It is evident from the foregoing discussion that there is a significant transfer of information between various levels of organization in such an integrated design system.

Adhoc Decomposition

The generation of structural topology was subject to constraints on the displacements at the nodes, natural frequencies of the structure, and the load carrying capacity of the individual members. Since the approach adopted in this work inherently examines several topologies, it was essential to use the constraint satisfaction as a factor to make the choice of a candidate topology from the available

set of topologies. To achieve this objective, an 'adhoc' decomposition procedure was used to satisfy the design requirements in a predetermined order. Alternate topologies were generated and analyzed when the initial topology failed to satisfy any of the structure behavioral constraints; a specific order was assumed for satisfaction of these constraints. The displacement constraints were considered first, followed by frequency, and finally the stress constraints. The implementation proceeded as a depth first search to identify the least weight topology that satisfied all the constraints, with set of topologies represented in a tree-like structure; the first level contains all topologies satisfying the displacement requirement, the second and third level with topologies satisfying dynamic and strength requirements, respectively. The satisfaction of the constraints need not necessarily follow the order suggested here. The problem of ordering the constraint satisfaction schedule is not as formidable as it may first appear, as a set of topologies were made available which helps in choosing a feasible topology, as detailed in the next few sections.

The topology synthesis proceeded until a satisfactory structure was synthesized. The initial task was to identify the least weight topology based on heuristics described in the previous section. Also, all other topologies in the order of increasing weight were generated in the event that a lower weight topology did not meet any of the design requirements. This leads to the generation of many possible topologies, including some which may be infeasible. Such a taxonomy gives the power to choose a suitable topology depending on the current problem requirements. This is essential as either a change in the order of decomposition (satisfaction of constraints), or a

change in any of the constraints during synthesis, may result in constraint violation in the chosen topology. However, the set of topologies generated for a given problem specification remains unchanged as these topologies are generated on the basis of parametric information for the particular design domain. In such situations, alternate topologies provide a greater scope for feasible topology identification.

Topology Analysis

The analysis of the structure was performed using a general purpose finite element program EAL. Displacement constraints were first checked for feasibility, the failure of which invokes a set of heuristics designed to obtain constraint satisfaction. These heuristics were based on incremental changes in the structure designed to incur the least weight penalty. An alternate structure is generally proposed by addition of a member at the node location for which the displacement constraint was violated. The member is connected to nearest support point to which it was not previously connected. Figure 3-1 depicts the topology refinement procedure that was employed. This process was repeated for a satisfaction of the frequency and stress constraints by invoking a similar set of heuristics.

The concept of backtracking included in the problem to identify the least weight topology, helps in managing different topologies that are a consequence of the adopted procedure. Simply stated, the backtracking strategy helps in choosing a least weight candidate topology, which satisfies all the design requirements (chapter 2).

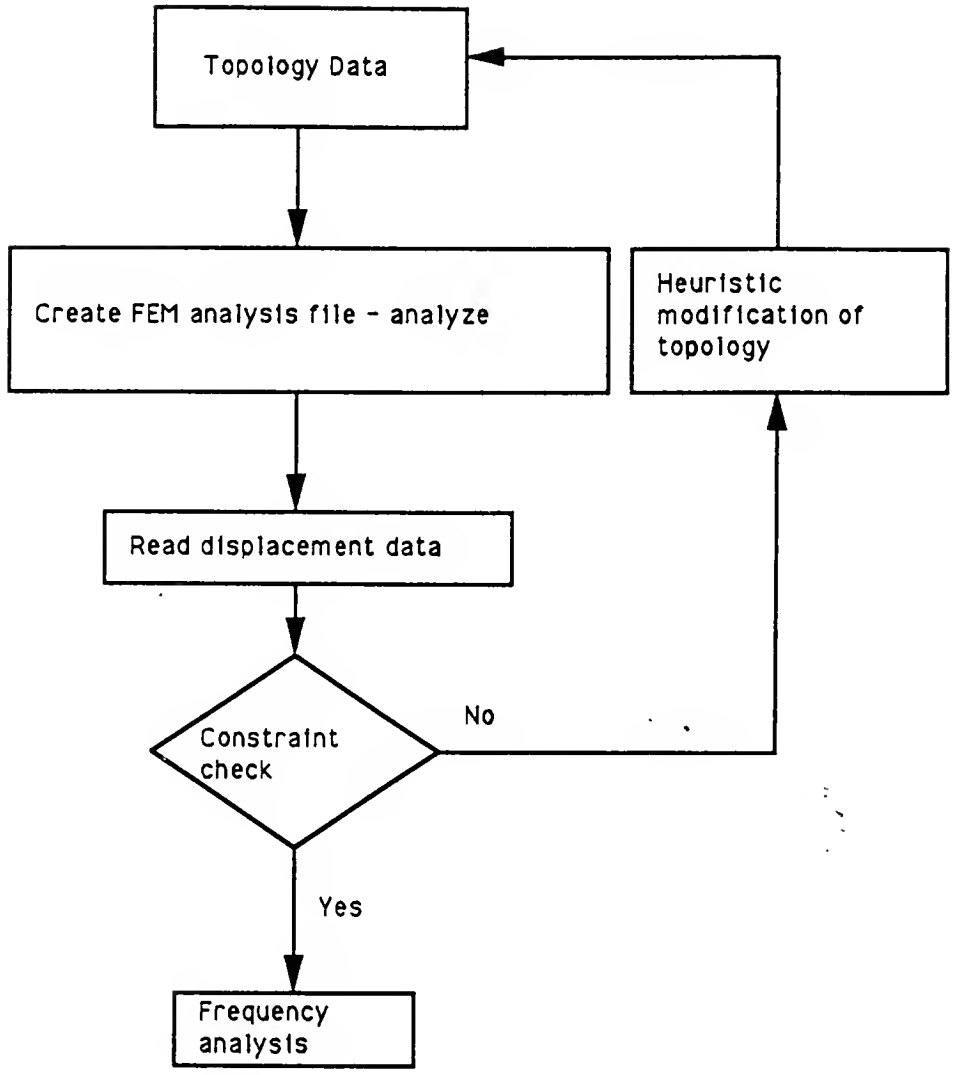


Figure 3-1. Topology refinement procedure.

In the event that after all possible connections the current constraint was still violated, the controller used a backtracking strategy (figure 3-2), to return to the previous constraint and look for the next best configuration acceptable for that constraint. This design was then passed to the immediate lower level to check for the satisfaction of the failed constraint.

The chosen topology was first checked for displacement constraint violation. If a violation was established, then the next heavier topology was chosen and displacement constraint verified. In the event that the topology passed the displacement criterion, then it was verified for the dynamic constraint. Successful validation of the dynamic constraint was followed by a verification of the stress constraint. In case, the dynamic constraint was violated, then backtracking procedure of figure 3-2 returned control to the previous level at which the displacement constraints were satisfied. Such a step by step analysis and backtracking was performed at three levels, with the next higher weight topology chosen if constraints were violated at any stage. Upon satisfactory completion of constraint satisfaction, the backtracking procedure was terminated and the candidate topology was passed on for subsequent optimization. A feasible structure may or may not result from the exercise. This would depend upon the extent of redundancy available in the problem domain. In the event of an infeasible design, one may either discontinue the exercise or attempt constraint satisfaction by varying member cross sectional properties.

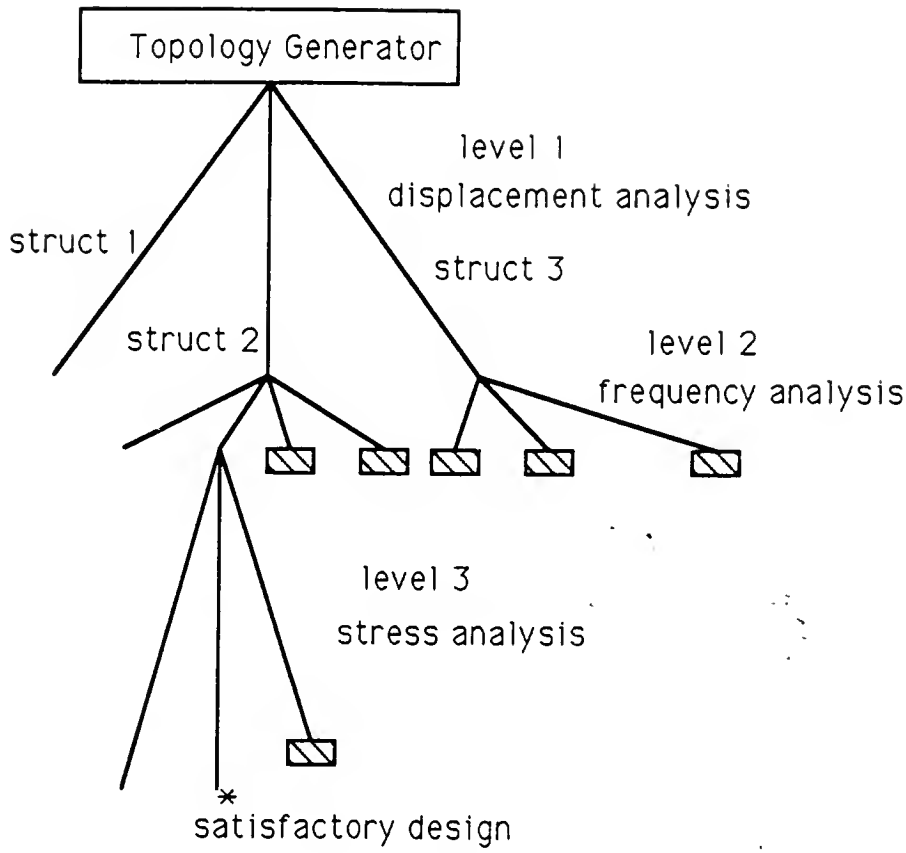


Figure 3-2. Backtracking strategy.

Heuristic Optimization

Once an initial topology has been generated, the next task in the optimal design process is the resizing of the cross sections of the structural members. In the present work, an attempt was made to reduce the extensive numerical evaluations required in an iterative procedural process, by incorporating some heuristics in the optimization procedure. The objective of this work was to simulate the search process of a constrained minimization scheme, but to reduce the computational effort by making gross heuristic assumptions regarding the suitability of a chosen search direction.

The sensitivities of the objective and constraint functions with respect to the design variables were first evaluated. Likewise, the initial values of the normalized constraints and the objective function were obtained as an assessment of the starting design. These were evaluated on the basis of a finite element analysis. The section areas were the design variables for the problem.

The design variables were ordered on the basis of the objective function sensitivities, with the first corresponding to the one that gave the best improvement in the objective function. The controller first attempted to satisfy any violated constraints, by changing the design variable that resulted in the least weight penalty. The actual step length was determined on the basis of a piecewise linearization of the violated constraint about the initial point.

Once an initially feasible design was established, the objective function and

constraint sensitivities were recomputed and ordered as before. While some of the constraints may be critical, others could be satisfied by large buffer margins, holding out hope for further decrease in the objective function. The design variable with the maximum possible improvement in the objective function was then chosen. Constraint gradients were checked to see if a change in this design variable could be accommodated without violating a constraint. If the piecewise linearization indicates the possibility of an infinite move in this direction, then a 25% change in the design variable was allowed. Otherwise, the constraint closest to critical for a move in this direction was checked to see if at least a 10% improvement was possible before the constraint was violated. If such an improvement was confirmed, then the allowable change was affected. Failure of this check resulted in examination of the next best available move direction. Note that if an unbounded move is suggested for a design variable in more than one iteration, it indicates a member that can possibly be removed from the structure, provided it does not render the structure unstable. It is worthwhile to note that such deletion of members is a difficult step in strictly procedural optimization.

Illustrative Examples

The specific organization of design tools in the context of the three level approach described in the previous chapter, is outlined in this section. At the function level, the topology analysis module was responsible for developing a table

of pertinent features for the loads and supports. Two other modules at the function level were programs for finite element analysis, and a nonlinear programming-based optimization program. The latter is available to perform procedural optimization on a given geometry in lieu of the heuristic optimization. Also present at this level was a controller which directed the flow of the synthesis process. In this implementation, the controller assumed an adhoc decomposition of the problem, and addressed the various design requirements in that order. A more rational approach of using global sensitivity analysis and dynamic programming concepts for this purpose is reported in later chapters. As stated in the previous chapter, no problem solving knowledge was introduced at the program level. However, the inference facility which uses information from the knowledge and function level to suggest new design moves, was available at this stage.

The approach presented in the preceding sections was applied to the generation of optimal topologies for 2-dimensional structural systems. Two problems, as described in figures 3-3 & 3-4, illustrate the proposed approach. The design domain contains both concentrated loads and moments, as well as forbidden zones in which no element may be introduced. The example of figure 3-3 used standard nonlinear programming algorithm [CONMIN] for member sizing optimization, and that of figure 3-4 was optimized on the basis of embedded heuristics for optimization. Tables 3-1 and 3-2 show the optimized areas for the two example problems, based on procedural approach and heuristics directed optimization. The difference in the weight of the optimized structure of figure 3-4 was 18% between procedural and

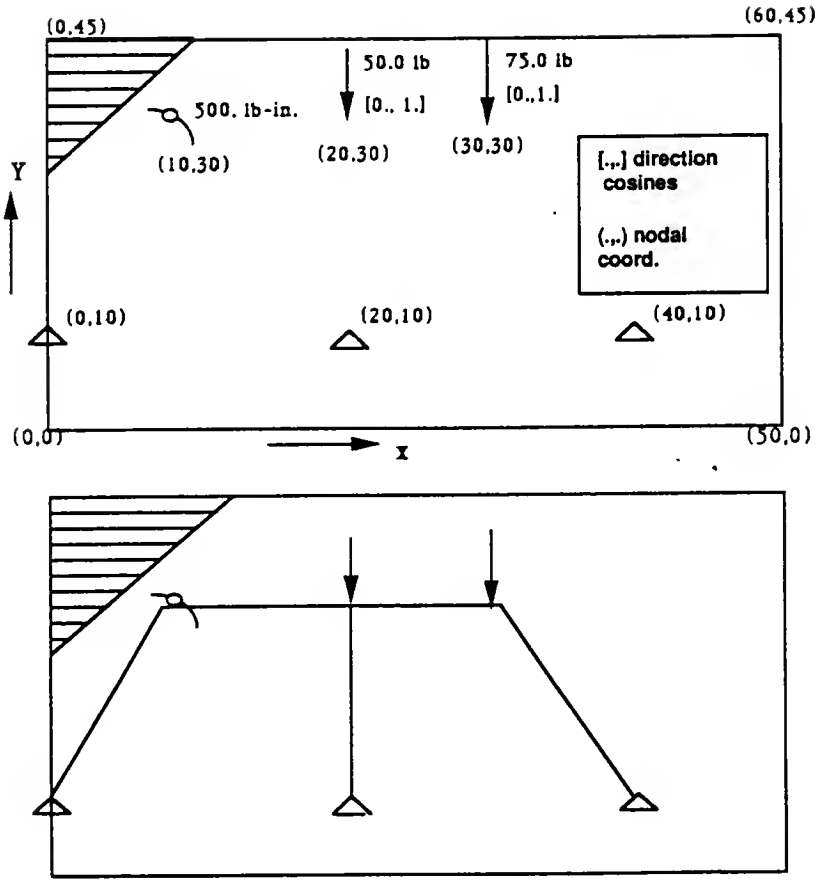


Figure 3-3. An example domain and proposed topology.

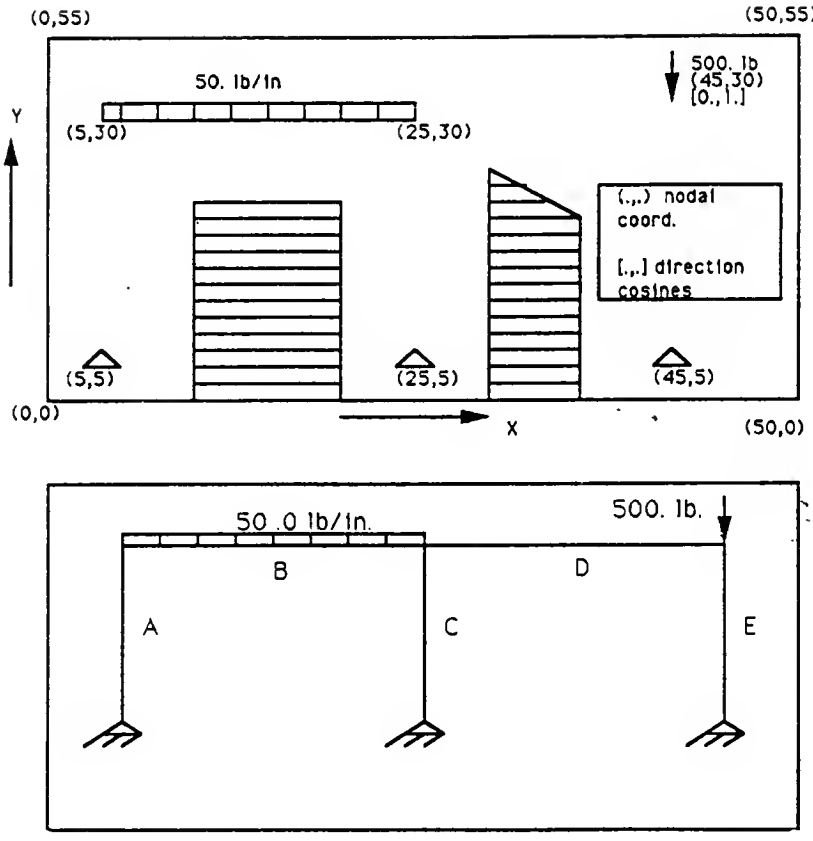


Figure 3-4. Problem domain with distributed loads.

Table 3-1. Optimized areas for topology in figure 3-3.

Member	Area (sq. in.)
1	2.52
2	3.80
3	3.80
4	4.53
5	4.53

Table 3-2. Optimized areas for topology in figure 3-4.

Element	By heuristics Area (sq. in.)	NLP approach Area (sq. in.)
A	5.6	4.53
B	4.5	3.9
C	3.1	2.97
D	5.8	5.1
E	3.3	2.65

heuristics based optimization, with the heuristics based optimization yielding the heavier structure. Figure 3-5 portrays the incremental refinement of the structural topology in example 1.

Topology Dependence on Order of Decomposition

For the problem shown in figure 3-3, the order of constraint satisfaction was varied to determine its influence on the final structural topology. Frequency constraint, stress requirements, and displacements were satisfied in that order, as opposed to previous order of decomposition. The topology obtained in this approach was the same as before, because the candidate topology corresponds to the least weight topology thus far synthesized, that satisfies all the three design requirements.

The design domain under consideration was simple enough so that the generated topologies were unaffected by the order in which the constraint satisfaction was introduced. The more general case would be one in which the topology would indeed be influenced by the order of constraint satisfaction. As an example, if there are two equivalent solutions to satisfying a particular constraint, choosing one over the other may effectively eliminate the possibilities of some topologies as other constraints are satisfied.

Similarly, the dependence of final topology on design requirements is also paramount. Any change in the design requirements alters the design space under consideration. If such changes are critical, different requirements may dominate the

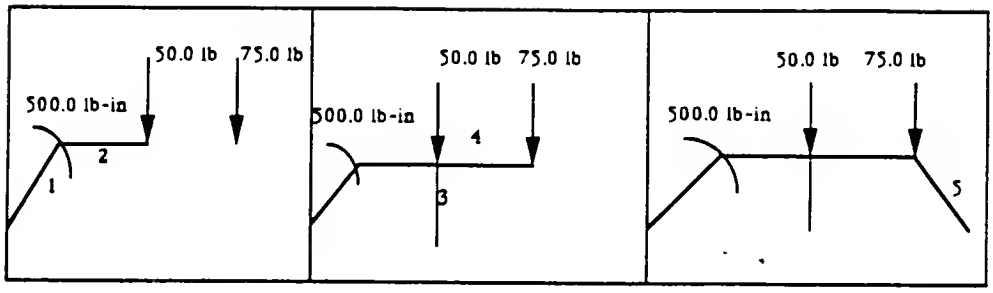


Figure 3-5. Incremental design synthesis.

development of topology. This is dependent on the amount of change in the design requirements, whether it causes a relaxation or tightening of the constraints.

CHAPTER 4
GLOBAL SENSITIVITY BASED DECOMPOSITION

Synthesis and design of complex structural systems is an involved process which must simultaneously consider a large number of possibly conflicting design requirements. The problem is difficult to handle by traditional mathematical tools which have been shown to perform very poorly in the presence of large dimensionality problems. Furthermore, in problems of topology generation, the design space is disjoint and is not amenable to standard solution techniques. Quasi - procedural methods present an alternative solution strategy for this class of problems. A look into the characteristics of the problem usually sheds light on some form of intrinsic coupling of characteristics of the problem. Such couplings, if identified, can be used as breakaway points, to alleviate the design process. Decomposition methods have been proposed to take advantage of such couplings. The aim of decomposition being to reduce the complex design problem into many simpler, preferably single goal problems.

Two methods of decomposition have been used in prior work. The first approach is the one presented by the 'adhoc' decomposition procedure described in chapter 2. That decomposition stipulated that the structure be designed satisfying the design requirements in some specified order. In such an approach, the design is dependent on the prescribed order in which the constraints were satisfied. A

second and more recent approach proposes the decomposition of complex systems into smaller subsystems to simplify the computation of coupled system sensitivity [24]. Although this decoupling may lead to many subsystems, the analysis in such systems may be conducted simultaneously. The final result of the analysis of all subsystems is a first order sensitivity of the behavior response, information that can be used during the optimization of the design. A third method for problem decomposition has its basis in dynamic programming methods. This research effort is discussed in chapter 5.

The present effort may be considered as a precursor to a design system for large structural systems. Here, a number of factors which influence the synthesis of a structural system are used to decompose the design domain into many relatively small design domains. The basic idea is to synthesize the partial structures in the subdomains, and to integrate these partial designs on the basis of a global sensitivity analysis. Once the partial structures have been integrated into a single structural system, members of the structure are optimized. Towards this end, both a heuristics based strategy and a procedural approach were implemented. Subsequent sections describe the design system implementation.

Decomposition Based Design Methodology

The design process is essentially quasi-procedural, wherein a design synthesized based on heuristics is validated using standard algorithmic procedures.

Figure 4-1 is a representation of the interaction of the various modules in the method. There are three main modules to be considered. These are synthesis, sensitivity analysis, and optimization modules, each one of which will be discussed in subsequent sections. The problem is described in terms of a design domain with concentrated loads, supports and forbidden zones where no structural members may exist. The aim of the design process is to evolve a minimum weight structure by decomposition of the problem domain, evolution of partial structures, and integration of the partial structures. Interaction of partial designs were taken into consideration through the use of global sensitivity analysis equations.

Figure 4-1 features the various processors embedded in the different modules of the design system. Procedural subroutines are invoked to evaluate the design domain, to determine the sensitivity analysis information and to perform gradient based optimization. The most important feature is the determination of the linking members required in the integration of partial structures. This is accomplished using heuristics, which in turn use procedurally developed data such as the sensitivity information for the structures under consideration.

Figure 4-2 depicts a typical design problem. In the presence of large number of such problem attributes, a topology assignment based on design heuristics cannot proceed in a rational manner. Hence, the first step in the proposed methodology was to decompose the problem domain into smaller subsystems, and to create partial designs for each of these sub-domains. The analysis and grouping of the parameters (loads, supports, forbidden zones etc.) was carried out to clearly identify the

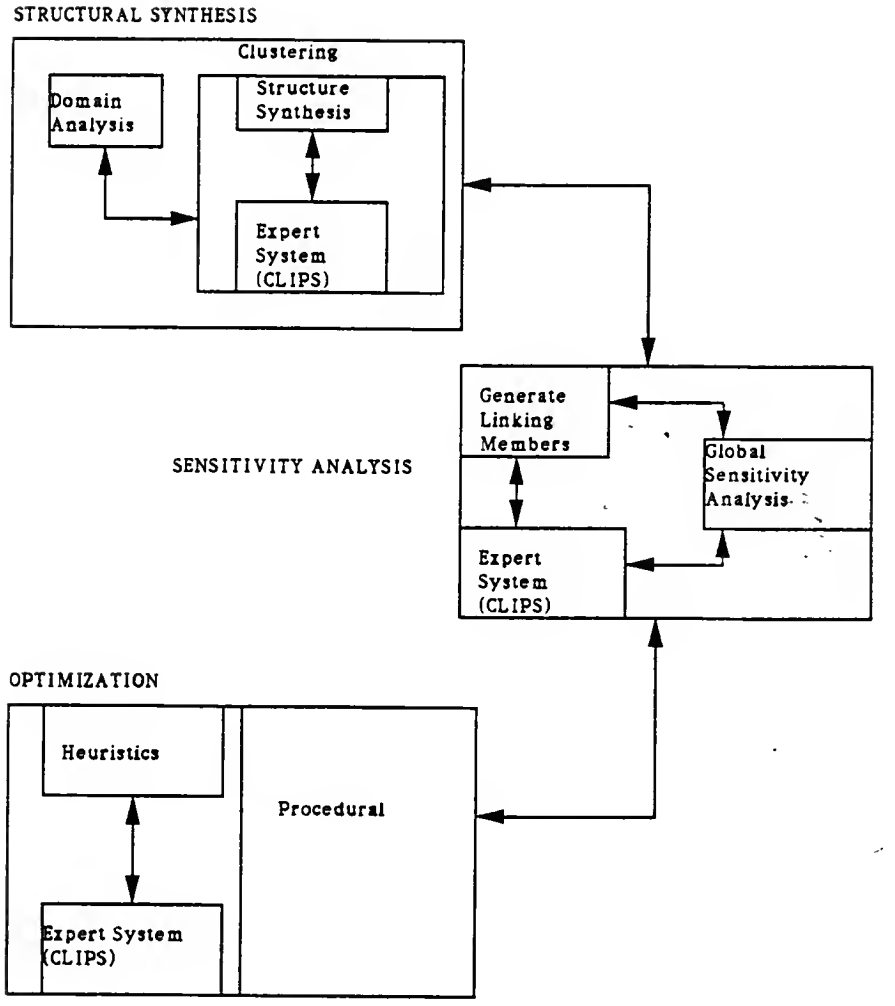


Figure 4-1. Different modules in the implementation.

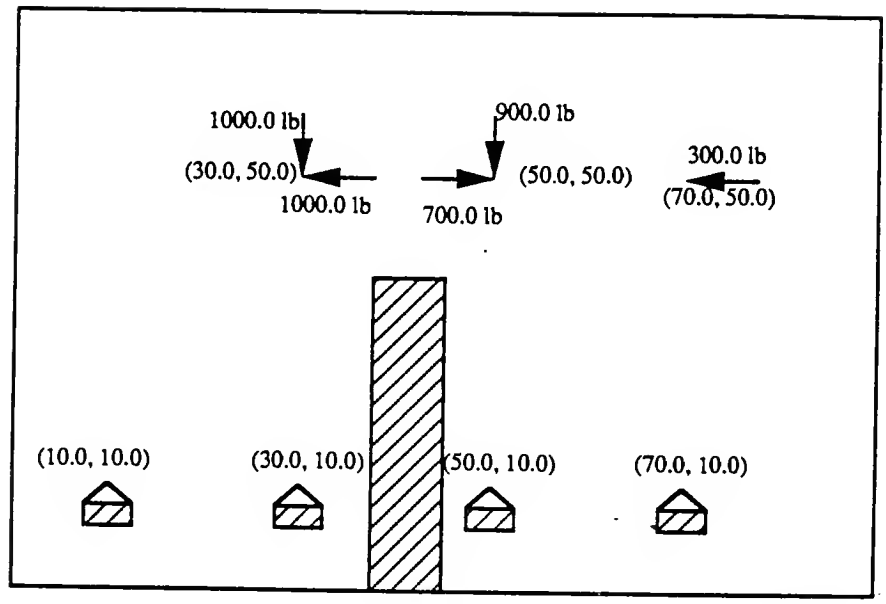


Figure 4-2. Example problem domain.

clusters(subsystems). Structures were then synthesized for each of the clusters. The next task involved the sensitivity analysis for structures generated for each of the clusters. In the present approach, these partial designs were also required to satisfy a subset of the design constraints for the problem. Most typically, these constraints were those that were influenced by the local variables. Global type constraints such as natural frequency constraints, were accounted for in the assembled structure. Assembly of the partial structures into a single structure is critical and requires careful consideration of the effect of coupling on both the local and global behavioral response of the system. A more detailed description of the process and its implementation are discussed in subsequent sections of this chapter. In all cases, the framework developed in chapter 2 was adopted to whatever extent possible.

Clustering

The parameters in a given design domain (figure 4-2) such as the load points, supports, types of loads and supports, etc. are characterized into a number of groups or clusters. Each group is considered as a design synthesis problem and a suitable structural topology is generated. This is the essence of the clustering process employed in the present research effort.

The division of the given domain into many subdomains is governed by geometry considerations; human designers approach such complex problems in much the same way, and use proximity of loads and supports to generate partial designs.

The method of clustering used in the current implementation is based on geometry and parametric information. The given design domain including any forbidden zones, was divided into quadrants, each containing a maximum of five loads and supports.

The mean distances are evaluated between loads and supports for the domain. The distances are measured along the line of sight, such that no member between any two loads or between a support and a load intersects a forbidden zone. If such an intersection occurs, a penalty is assigned by increasing the computed length substantially, thereby rendering its selection unattractive from the standpoint of increased weight. A vector V_L with components consisting of mean distances between loads, mean distances between loads and supports, number of loads, and number of supports, is generated for each load point in the domain.

$$V_L = \begin{Bmatrix} \dfrac{\sum d_L}{n_L} \\ \dfrac{\sum d_{LS}}{n_{LS}} \\ n_L \\ n_S \end{Bmatrix} \tag{4-1}$$

Here d_L is the load-load distance, d_{LS} is the load-support distance, n_L is the number of loads considered, and n_S is the number of supports considered.

Clusters are formed on the basis of the vector of characteristics described in (4-1), using the following heuristics. Any cluster may not contain more than a total of five parameters. In addition, it was necessary to include at least one support and

one load point in each cluster. Within a cluster, the shortest load-to-load and load-to-support distances are preferred, as they are most likely to yield a lower weight configuration. Finally, and to whatever extent possible, an average number of parameters is maintained for all clusters. Given that the maximum number of parameters must not exceed five, and the total number of parameters known, a decision can be made on the number of clusters and on the number of parameters that each must contain. An arbitrary decision was made to begin generating clusters from the load located closest to the origin of the coordinate system for the design domain. The load closest to the first load, but not included in the first cluster was the starting point for the second cluster. This pattern was repeated until all loads and supports were assigned to some cluster. The only binding factor in limiting the number of parameters in any cluster is the guarantee of a simple structural topology within the cluster. Once a cluster was established, a cluster center was identified. The center for each cluster was determined by taking the average of the position coordinates of all parameters in the cluster. The mean distances of all loads and supports from the center of the cluster, were evaluated for each cluster. Figure 4-3 depicts the clusters for the example problem shown in figure 4-2.

Also important to each cluster are the I/O points at which all relevant load transfers are to be monitored. It is at these points that the influence of an adjacent cluster will be transferred to the cluster under consideration. The choice of any node as an I/O point is based on its proximity to the adjoining cluster boundary. Nodes

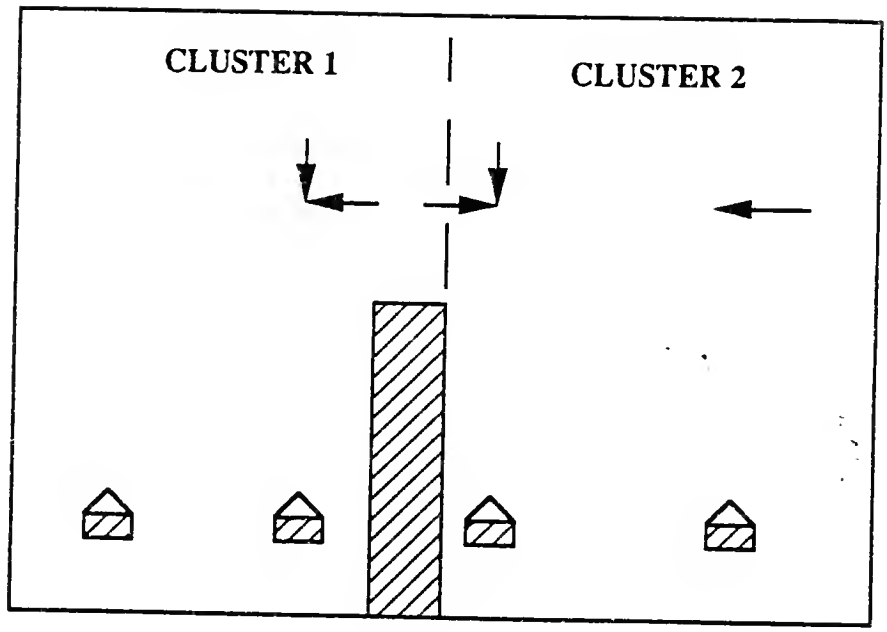


Figure 4-3. Clustering for the example problem.

which lie in close proximity to adjoining cluster boundary are all considered I/O points. These points are very important since the information available at these points such as loads and displacements for each partial design, have a direct bearing on the introduction of the connectivity elements between clusters.

Partial Structure Generation

The successful clustering of all input parameters leads to the next important stage in which a topology is actually generated for each cluster. All the parameters (both loads and supports) that fall within a cluster are recognized and accounted for. Such parameters along with their pertinent information are then passed into a subroutine which prepares an output file, to be used in conjunction with previously developed knowledge based structural synthesis programs detailed in chapter 2. This synthesis is sequential, with each load stabilized one at a time. Once a load is stabilized, the point of application of this load serves as additional support point. Additionally, nodes are introduced when the shortest distance between some load and available connection points becomes unacceptably large, along the line of sight. An example of this could be a situation where a long member were to become critically unstable under a compressive load. A suitable structure is generated based on the parameter information within each cluster, and this process is repeated for all clusters. The partial designs generated for the problem are depicted in figure 4-4.

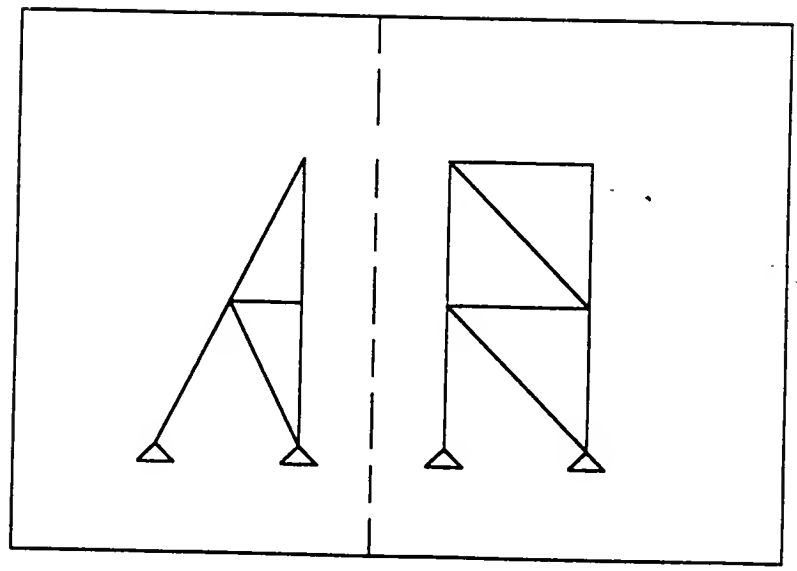


Figure 4-4. Partial structures.

Inter-Cluster Connectivity

Once all the partial designs are available, the next objective is to identify the possible I/O points and the design of corresponding connecting elements. The approach adopted was to consider only adjacent clusters for connectivity as shown in figure 4-5. Nodes located close to the cluster boundary as measured from the cluster center, and also in close proximity to the adjacent cluster were tagged as possible I/O nodes for inter-cluster connections. Connection members of nominal cross section were introduced from I/O points in the first cluster to all the I/O points in the adjacent clusters. The I/O points in adjacent clusters were treated as simple supports, and the reactions at these supports were determined for the applied loads in the first cluster. Figure 4-6 shows the connectivity elements for the problem under consideration.

The integration of partial structures generated for all the clusters, to form a single structure, constitutes the next step in the design procedure. At this stage, global sensitivity analysis is performed to determine the influence of the design variables in any partial structure with regards to the load effects introduced during integration with other partial structures.

Global Sensitivity Analysis.

A description of the method of global sensitivity analysis can be found in

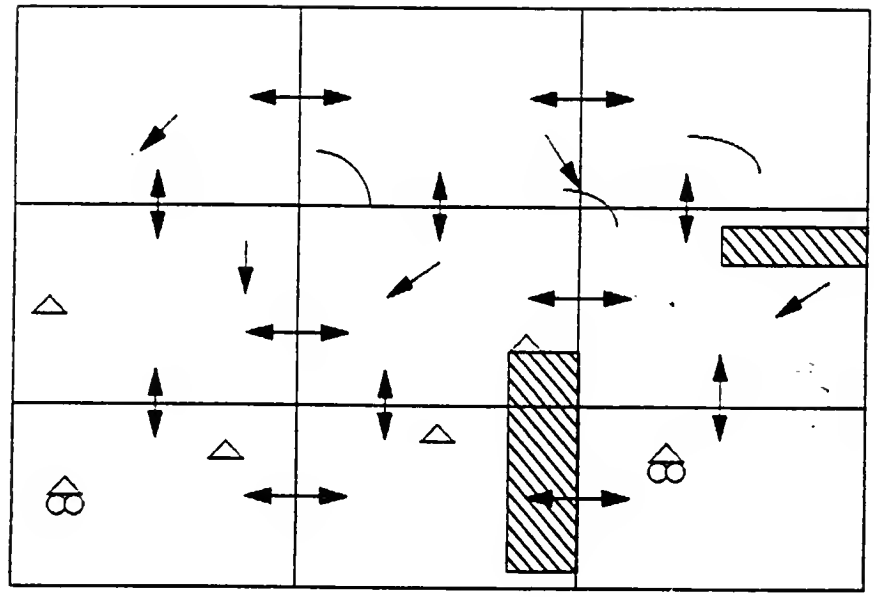


Figure 4-5. Allowable connectivities between clusters.

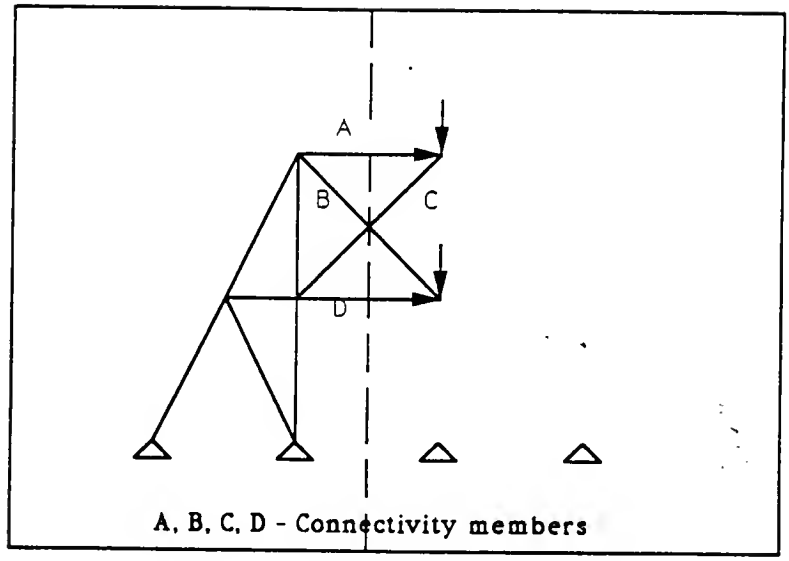


Figure 4-6. Possible elements of connectivity.

Sobieski [24]. However, a summary of the approach is presented here for completeness.

A generic coupled system is represented schematically in figure 4-7. The system consists of two participating subsystems A and B. Complete coupling is present as the output of one subsystem influences the output of the other. The functional analysis equations for these two disciplines can be represented as

$$((X_A\ ,\ Y_B)\ Y_A) = 0 \tag{4-2}$$

$$((X_B\ ,\ Y_A)\ Y_B) = 0 \tag{4-3}$$

Here X's are the local or intrinsic variables, and Y's (also referred as extrinsic variables) are their respective outputs. Some of the Y's comprise the interlinking parameters between the subproblems as shown in figure 4-7. The total derivative of the output of each subsystem with respect to their intrinsic variables are obtained on the basis of first order Taylor series expression as follows:

$$\frac{dY_A}{dX_A} = \frac{\partial Y_A}{\partial X_A} + \frac{\partial Y_A}{\partial Y_B} * \frac{dY_B}{dX_A} \tag{4-4}$$

$$\frac{dY_B}{dX_B} = \frac{\partial Y_B}{\partial X_B} + \frac{\partial Y_B}{\partial Y_A} * \frac{dY_A}{dX_B} \tag{4-5}$$

The cross sensitivities were derived using chain rule.

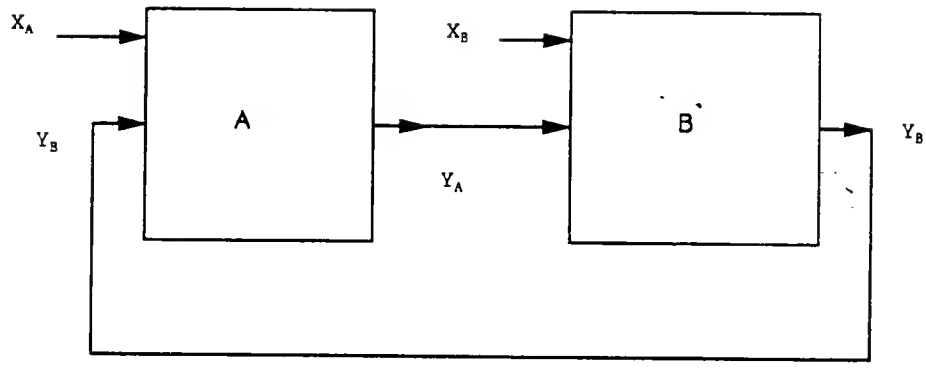


Figure 4-7. An intrinsically coupled system.

$$\frac{dY_A}{dX_B} = \frac{\partial Y_A}{\partial Y_B} * \frac{dY_B}{dX_B} \tag{4-6}$$

$$\frac{dY_B}{dX_A} = \frac{\partial Y_B}{\partial Y_A} * \frac{dY_A}{dX_A} \tag{4-7}$$

These equations can be written in a compact matrix notation as follows:

$$\begin{bmatrix} I & -\frac{\partial Y_A}{\partial Y_B} \\ \frac{-\partial Y_B}{\partial Y_A} & I \end{bmatrix} \begin{bmatrix} \frac{dY_A}{dX_A} \\ \frac{dY_B}{dX_A} \end{bmatrix} = \begin{bmatrix} \frac{\partial Y_A}{\partial X_A} \\ 0 \end{bmatrix} \tag{4-8}$$

$$\begin{bmatrix} I & -\frac{\partial Y_A}{\partial Y_B} \\ -\frac{\partial Y_B}{\partial Y_A} & I \end{bmatrix} \begin{bmatrix} \frac{dY_A}{dX_B} \\ \frac{dY_B}{dX_B} \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{\partial Y_B}{\partial X_B} \end{bmatrix} \tag{4-9}$$

It is important to note that the total (coupled) derivatives of the output of each subsystem with respect to the intrinsic variables of all subsystems depends on the partial derivatives which may be obtained within each subsystem, and for some nominal value of the outputs of other subsystems. These sensitivities can be used later in the optimization process.

Design Constraints. In the design problem under consideration, system A can be viewed as the partial structure in a cluster, and system B used to denote all possible linking members between two adjacent clusters. Each cluster is considered individually and similar subsystems are identified. In the equations derived before, X_A's are the cross sectional areas of the members in a structure for cluster #1; Y_A's are the outputs for cluster #1, which are a cumulative measure of constraint satisfaction/violation for that cluster, and the weight of the cluster structure; Y_B's are the weights (W_B) of the interlinking members, and the output reactions which act at the I/O node in adjacent cluster; X_B's are the cross sectional areas of the interlinking members. Similar inputs and outputs are identified for cluster #2.

The output from the analysis of any structure consists of the sensitivities of the cumulative constraint, the weight of the structure, and reactive loads at I/O points in the adjacent cluster, with respect to all design variables. The cumulative constraint as used in the present work is taken from Hajela [25] and is written as follows:

$$\Omega = -\epsilon + \frac{1}{\rho}\ln\left(\sum_{i=1}^{m} (e^{\rho g_i})\right) \qquad (4\text{-}10)$$

In the above equation g_i's are the m stress and displacement constraints for each cluster, ϵ is of the order of 10^{-3}, and ρ is a parameter taking values between 20 and 30. The effect of the Ω function is to create an envelope function representative of all the m individual constraints.

A finite element analysis is performed on the partial structure, including a structural member introduced between chosen I/O nodes on the partial structure and the structure of an adjacent cluster. The I/O nodes in adjacent clusters are considered to be simple supports and the reaction forces computed at these nodes. All such combinations of I/O nodes are considered. The total sensitivities of the structural weight, cumulative constraints and the reaction forces at I/O points of adjacent cluster are determined as follows:

$$\left\{\frac{dY_A}{dX_A}\right\}^T = \left\{\frac{d\Omega}{dX_A}, \frac{dW_A}{dX_A}\right\}$$

$$\left\{\frac{dY_A}{dX_B}\right\}^T = \left\{\frac{d\Omega}{dX_B}, \frac{dW_A}{dX_B}\right\}$$

$$\left\{\frac{dY_B}{dX_A}\right\}^T = \left\{\frac{dW_B}{dX_A}, \frac{dF}{dX_A}\right\}$$

$$\left\{\frac{dY_B}{dX_B}\right\}^T = \left\{\frac{dW_B}{dX_B}, \frac{dF}{dX_B}\right\}$$

Reaction forces, and sensitivities of the cumulative constraint, weight and reaction

forces, determine those pairs of I/O nodes between which connecting members must be introduced. This is done on the basis of heuristics embedded in a knowledge base, and is discussed in the following section.

Heuristics for Assembly of Structures. Pertinent data utilized in the determination of candidate connectivity elements include all the global sensitivities, the reaction forces, and displacements at various nodes in the structure. Connectivity members are evaluated one at a time, until all the connectivity members are accounted for. The objective is to choose that member which promotes constraint satisfaction or introduces the least constraint violation, while still minimizing the structural weight penalty associated with that member. This translates quantitatively into choosing a member which has the least total sensitivity value of the cumulative constraint with respect to the cross sectional area of the member under consideration, provided it also carries the least weight penalty. Towards this end, the sensitivity of the structural weight to such an addition is obtained. The other measure of merit in choosing a member is the dependence of the reaction forces at I/O points in the adjacent cluster, on the member cross section. Such sensitivities are compared for each possible connectivity member, and the hierarchy of comparison is the sensitivity of the cumulative constraint, weight, and reaction forces, in that order. A member which provides a minimum for all the three sensitivities, or most components of the sensitivity as per the specified hierarchy is selected as the connectivity element.

Implementation

The decomposition method was implemented in the general framework of a design system described in chapter 2. All algorithmic analyses was performed in FORTRAN, the heuristics were implemented using the CLIPS syntax, and structural analysis performed using EAL.

The approach was applied in the generation of topologies of representative structural systems as illustrated by the example. Figure 4-8 portrays the complete structure for the example problem.

The identification of the connectivity members leads to the assembly of all partial designs. Before the integration of partial designs, the members of each individual structure in each cluster, are optimized for minimum weight. The elements obtained by this cycle were fixed, and the procedure repeated for all clusters. The total sensitivities are utilized in the optimization of the structures. Two different methods of optimization were performed to obtain optimal design. The first method involved the use of heuristic optimization routines as reported in chapter 2, while the second was based on traditional procedural methods for optimization. The structures were optimized subject to the respective cumulative constraint. The individual constraints for each structure were those introduced due to local nodal displacement and element stress considerations. The stress constraint on the connectivity member is considered as a local constraint. The connectivity element is considered as an integral part of the structure in a cluster, and the I/O node

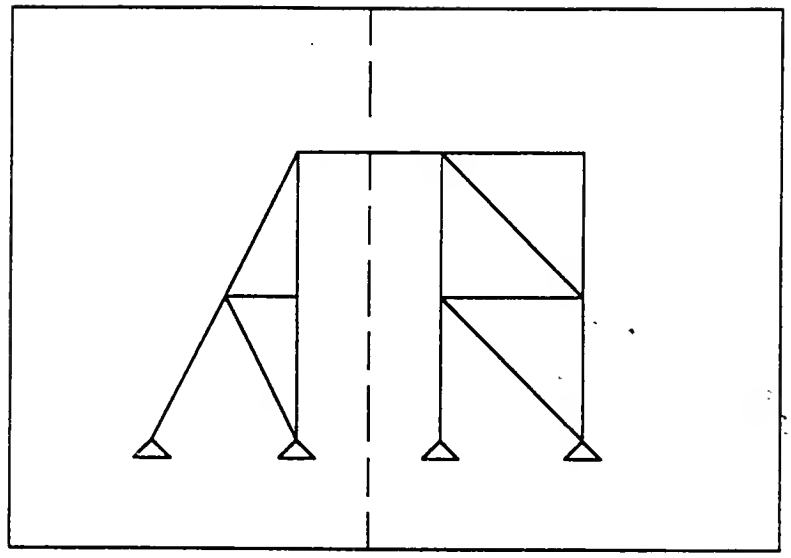


Figure 4-8. Complete structure.

treated as a simple support, as described in an earlier section. The integrated structure continually evolves until all clusters have been taken care of. Note that in each design cycle, the number of design variables are only governed by the number of members for that cluster. Finally, the integrated structure can be optimized to accommodate any constraint of a global nature.

Procedural optimization was performed using feasible-usable search algorithm in ADS[26], and was executed in a batch mode by the design system.

CHAPTER 5
STAGEWISE PROBLEM DECOMPOSITION

In the previous chapter, the concept of decomposition as a solution to complex design problems was approached from a global sensitivity standpoint. In this chapter, decomposition is based on concepts of dynamic programming, which are typically applicable to those design problems that may be considered as a series of subproblems. The nature of the structural synthesis problem considered here may be thought of as a serial process, wherein the output from a subproblem is considered as an input to the succeeding subproblem. Therefore, it is a prime candidate for adaptation of the dynamic programming approach. The nature of the formulation transforms a problem into a different form, frequently more suitable for solution. Reported efforts in the application of dynamic programming to structural design are somewhat limited. Kirsch [27] provides insight into the application of dynamic programming to affirm the topology of an existing structure. This publication also cites other rudimentary applications, such as design of cantilever beams.

A system with distinct subproblems which is organized such that a change in the design of a given subproblem influences only downstream subproblems, can be treated as a serial process. DP is based on a theory proposed in the original work by Bellman [28] (pp. 12):

> An optimal policy has the property that whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Dynamic programming is useful when the decision sequence is long, and the number of decisions is considerably large. The basic feature is that at each given subproblem, the decision is affected only by the variables and functions present in that subproblem, and is insulated from the other subproblems.

Two different approaches were taken for structural synthesis and are reported in the following sections. The class of structural design problems under consideration were transformed into series of serial stages, so that principles of dynamic programming could be applied. Once a minimum weight topology was identified, the topology was optimized for minimum weight using genetic algorithms. A second approach which is the central theme of the next chapter, involves a genetic algorithm based methodology for topology generation, where all the candidates for the initial population are derived from a dynamic programming approach. Finally, the successful topology is optimized using genetic algorithms. The following sections outline principles of dynamic programming, genetic algorithms, and their adaptation in problem decomposition. Also reported is an illustrative example of the proposed implementation.

Dynamic Programming(DP)

Figure 5-1 shows a serial multistage system, for which dynamic programming

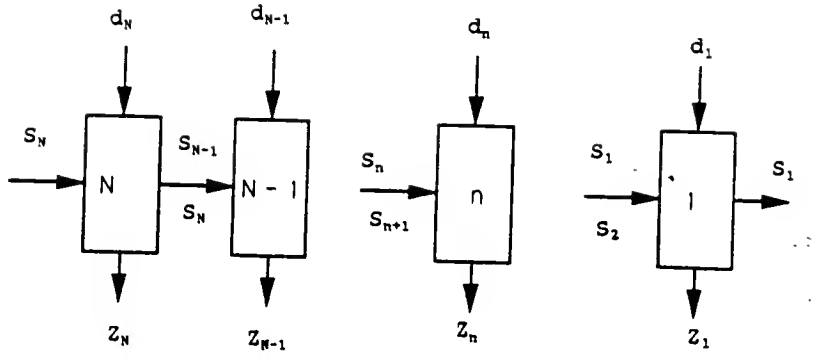


Figure 5-1. Multistage serial system.

can be applied. Since dynamic programming involves the execution of subsystems in a specific order, it is convenient to number the subsystems. The following terms and definitions are used in the formulation of a dynamic programming problem.

'S_n' state variables are those which carry information from stage to stage. The variable 'n' is referred to as the stage number. The input to nth stage is also the output from stage n-1. Therefore, we have

$$\tilde{S}_{n+1}=S_n \qquad n=1,2...N-1 \tag{5-1}$$

The functional relationship refers to flow of data between the subproblems (stages).

'd_n' decision variable at stage n, is an input variable which supplies input information to the nth subproblem. All the input variables which are not stage variables are generally referred to as decision variables.

'n' system stage is the subsystem where transformation of all stage variables and decision variables take place in accordance with functional relationships between them. The output state variable S_n is related to the input variable S through the transformation function, and can be represented as

$$S_n=T_n(\tilde{S}_n,d_n) \tag{5-2}$$

'Z_n' stage cost measures the objective function value at stage 'n'. The stage cost is a function of the input state and decision variables, and is represented as

$$Z_n = F_n(S_n, d_n) \tag{5-3}$$

The total objective is the summation of all individual objectives in all the stages

$$Z = \sum_{n=1}^{n} F_n(S_n, d_n) \tag{5-4}$$

The above definitions and relationships constitute a typical DP formulation. An adaptation to our problem needs some modifications as far as the variables are concerned, and this is described in the following section.

DP Adaptation in Stagewise Decomposition

The application of the DP like method to the problem of structural synthesis is largely dependent on the definition of the problem. The DP like approach is not particularly applicable to structural synthesis if it involves a routine optimization. Since DP is primarily a method of serial optimization, its adaptation will not be satisfactory in case of nonlinear design problems such as in structural member sizing.

Identification of a serial structure is the first phase. This involves the creation of subproblems as required in establishing a serial architecture. In the case of structural synthesis problems considered here, the adaptation was viewed from the point of individual load stabilization. The basic approach was again based on stabilizing each load, taken one at a time. The load stabilization task was considered

as 'n' distinct subproblems, with each subproblem having an objective to stabilize an individual load in all possible configurations, by introducing nominal cross sectional area members. The problem is propagated in this manner from one subproblem to the other, until all the loads are stabilized, simultaneously carrying forward the optimal weight (cost) thus far produced. The final topology at the end of the n-th subproblem is the one which yields the lowest weight. This topology was then optimized by resizing the member cross sections, subject to strength, displacement, and frequency constraints.

The basic design problem considered here is the same as reported in the previous chapters. The objective is to design a minimum weight topology, given a set of loads, possible support points, and domains where no members are allowed. Heuristics directed load stabilization approach (chapter 2) is adopted to transmit each load to the support points.

In accordance with the definitions reported in the previous sections, variables can be associated with the structural synthesis problem. Each load is considered as a 'stage' in the design process. All possible supports to which each load can be connected are considered as 'state' variables for that stage. In addition, all previously stabilized loads in the upstream stages, are also considered as 'state' variables. The 'cost' function for each stage is the weight of the partial structure for each stage. Details of the adaptation of DP to our problem, are described in the next section.

Implementation

The automated design procedure is implemented under the guidance of the architecture in chapter 2. Initially, as before, the design domain is enumerated and data retrieved pertaining to load-support and load-load distances. If such a connection intersects a prohibited zone, a weight penalty is associated with the connection. A topology knowledge base (chapter 2) drives the decision making process to stabilize each load individually. Nominal cross section elements are introduced when a load is stabilized. In the process all possible configurations are derived and their cost function in terms of weight evaluated. Such evaluations are performed for all stages.

The process of information transfer from an upstream stage to a downstream stage as defined by stage function, consists of the optimal cost functions for each state in the immediate upstream stage. Figure 5-2 depicts the flow of information from an upstream stage to a downstream stage. The '*' indicates the optimal value as transmitted from the upstream stage. At each stage in the subsystem, the optimal cost function is given by the equation

$$f_N(S_N) = \min(F_N(S_N) + f_{N-1}(S_N)) \qquad (5\text{-}5)$$

where 'f' is the optimal cost, 'N' is the current stage, and 'F' is the cost (weight) for the current state. Therefore, the optimal cost function is cumulatively updated during the progress of the analysis of subsequent stages. Ultimately, at the end of

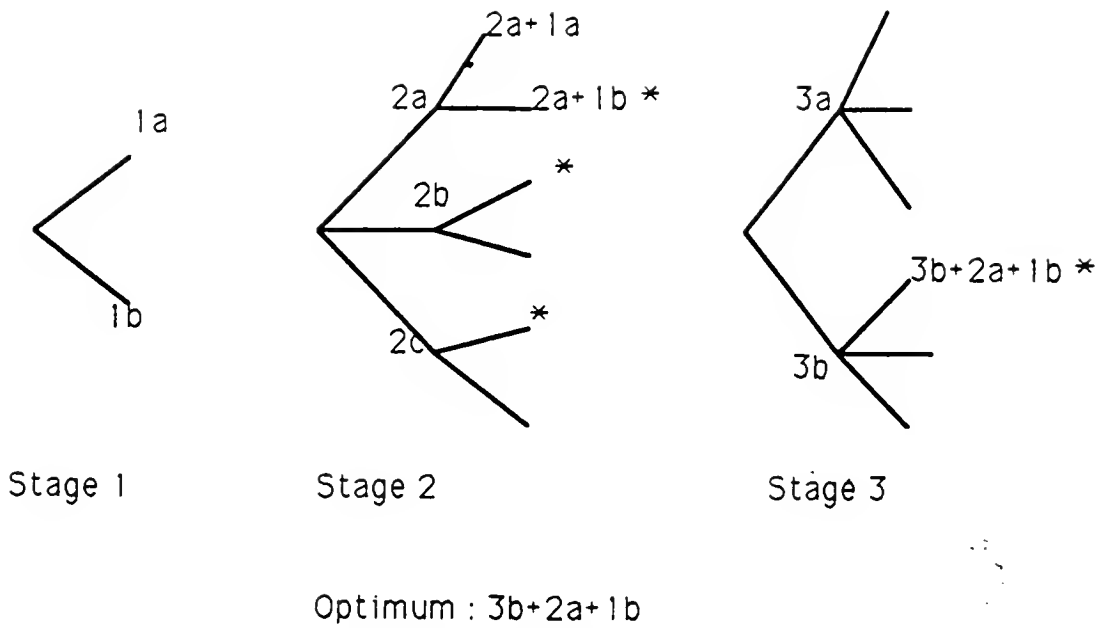


Figure 5-2. Information flow between stages.

the nth stage, after all the loads have been stabilized, the topology with the least weight is identified. At this point, the analysis is backtracked to identify the states corresponding to each of the upstream stages which contributed to this optimal solution.

The most important advantage in the application of DP to design synthesis is that all infeasible designs are pruned at each stage in a methodical way. Paths which hold promise are only allowed to propagate to succeeding stages.

An example problem that was considered for design synthesis based on the DP approach is shown in figure 5-3. Clearly, the number of stages can be identified as 4, which correspond to the total number of loads to be stabilized. The stabilization proceeds from load nearest to the origin of the coordinate system and moving away from it. All previously stabilized loads are also considered as possible supports. Hence, the number of states keep increasing with stages in the downstream. For the problem considered here, more than fifty topologies are possible, but the pruning of infeasible paths after the analysis of each subsystem reduces the available paths to a manageable number (24). Therefore, combinatorial explosions in the topology generation process is checked due to the pruning that is introduced at each stage. To further demonstrate the pruning of alternatives, let us again consider the example in figure 5-2. The number of stages for the problem is three. The three states in stage 2, combined with two possibilities from stage 1, results in six possible outputs. But since only optimal states are considered for propagation, only three states are identified, one each corresponding to each of the three states at stage 2. Similarly,

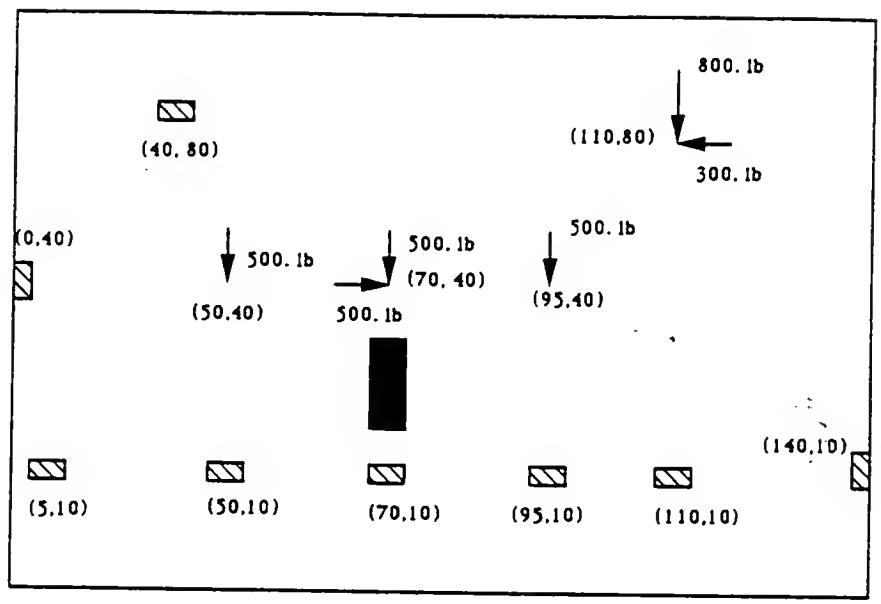


Figure 5-3. Example design problem.

there are two states at stage 3. Correspondingly, for each of the states from stage 2, there are two possible outputs from stage three, with a total of six. However, the actual number of outputs from stage 3 would have been twelve if all the possibilities in stage 2 had been considered. The use of this process, therefore, alleviates the combinatorial explosion in such problems.

The topology corresponding to the least weight structure for the example problem, is the focus of optimization using a genetic algorithm [29, 30, 31] outlined in the next section.

Genetic Search

Genetic search has its basis in Darwin's theory of evolution. It belongs to the general category of stochastic search techniques. A population consisting of design alternatives are allowed to follow the principles of evolution, such as to reproduce and cross among themselves, with a bias towards more fit members of the population. Combination of most fit members in the population results in a progeny population that is generally more fit than the parents. A measure related to the relative merit of members in the population serves as the objective function. There are three basic operations which constitute a genetic search process, namely selection, crossover and mutation, and these are explained in the next section. Representation of the evolution process itself requires completion of three preliminary steps. The first is a definition of a bit string representation scheme to portray possible candidate

solutions to the problem. Second, a suitable fitness function to provide for the ranking of the members in the population based on their merit. The last and most critical is the development of transformations to duplicate the evolution process consisting of the above mentioned operations.

Each design variable is represented as a fixed length string of 0's and 1's, typically the components of a binary number. A scaling scheme provides for the conversion of the binary number to real number and vice-versa. The binary strings for each design variable are placed end-to-end; this consititutes the string for one design. Several such designs constitute a population, with each member associated with a fitness measure. This is a measure of the relative goodness of the design, and is a composite measure of the objective function and the constraint satisfaction.

Basic Operations. Once a population of designs becomes available, then selection is the first concept to be introduced. The aim of this process is to select only those candidate designs based on their relative fitnesses, which are better than some others. This is meant to provide bias in the population to contain more fit members and to rid the population of less fit members.

The second component of the search is crossover, and pertains to the exchange of characteristics between select members of the population. Crossover entails selecting a start and end position on a pair of mating strings at random, and exchanging the 0's and 1's between these positions on one string with that from the mating string. This is illustrated as follows:

parent1 = 111000100101

parent2 = 010111001001

child1 = 111011001101

child2 = 010100100001

Mutation is the third transformation operator in the genetic refinement process, and safeguards the premature loss of valuable genetic material during selection and crossover. In the implementation, this corresponds to selecting few members of the population, determining at random a location on the string and switching the 0 or 1 at this location.

The above steps are repeated for successive generations of the population until either convergence is achieved on the fitness function, or a predetermined number of evolution cycles have been completed. The approach has shown promise in the current application because of its ability to search in a discontinuous design space where structural members can be either removed or introduced during optimization.

Optimization. The topology obtained from the DP method is optimized using genetic search. In this implementation the constraints were derived from displacements, natural frequency and stress requirements. The objective is to minimize the weight of the structure subject to the above constraints. A cumulative function with penalties assigned to the constraints was considered as the objective

function, and is given by

$$z = w + R * \sum < g_i >^2 \qquad (5\text{-}6)$$

where 'w' is the weight, 'R' is a penalty parameter such that the second term on rhs in (5-6) is of the same order of magnitude as the weight, and $<g_i>$ represents the normalized constraints and is defined as follows:

$$< g_i > = [g_i, 0] \quad if\ [g_i > 0, g_i < 0] \qquad (5\text{-}7)$$

Genetic search maximizes a prescribed fitness function. Since our problem is one of function minimization, a transformation of the objective is necessary. This is as follows:

$$z^* = z_{max} - z \qquad (5\text{-}8)$$

where z_{max} is the maximum value of z in the given population. The value z^* can serve as the fitness function for the population.

Initial population consisted of twenty different topologies, with cross sections of the members in the structure considered as design variables. All the designs in the population were randomly generated. A nine bit string representation scheme was chosen to represent each individual cross sectional area. A finite element analysis of each of these designs yields the values for the displacements, frequency, and member stresses. Constraints were evaluated and the fitness function

determined using the equations in the previous section. Since the population is randomly generated and is limited in size, if raw values of the fitness function are used in reproduction, the more fit designs dominate the population and the process may converge prematurely. Hence, the raw fitness function values were scaled appropriately to prevent premature convergence. Also, in the event that the average fitness is close to the maximum fitness, the scaling magnifies the difference in relative merits between members. The scaling used in the present implementation was linear, and was defined as follows:

$$f' = c_1 * f + c_2 \tag{5-9}$$

The scaled maximum fitness is given as $k*f_{avg}$, where k is typically in the range of 1.0 to 2.0. For a chosen value of k = 1.5, the c's in the above equation are derived from

$$c_1 = \frac{0.5 * f_{avg}}{f_{\max} - f_{avg}} \tag{5-10}$$

$$c_2 = \frac{f_{avg}(f_{\max} - 1.5 * f_{avg})}{f_{\max} - f_{avg}} \tag{5-11}$$

where f_{avg} is the average fitness for a given generation.

Scaled fitness values f' were used to detect the most fit members of the population for reproduction. Simulations of weighted roulette wheel were used for such a selection, where each member of the population occupied a sector on the

wheel in proportion to its scaled fitness value. This scheme was then used to select the mating members of the population. In any population, if one can identify any particular pattern in a string of length (L), for example

H(1) = ***10**0****

where H is the pattern, and * indicates that the position can be occupied by either a 0 or a 1, then we can predict the number of occurrences of the same pattern in the succeeding generation [32]. In any schema H, an important parameter is a defining length (δ), which is the distance between the first and last specific digit in a string; for H(1), δ is (8 - 4) = 4. Also important is the number of specific digits in any schema H, and is referred to as the order O(H); the order of H(1) is 3. If m(H,t) represents the number of occurrences of the pattern in generation 't', then the number of strings with the same pattern in the next generation is given by

$$m\ (H,t+1) = m(H,t) * \frac{f(H)}{f_{avg}} \qquad (5\text{-}12)$$

where f(H) is the fitness for the pattern. It is easy to conclude from the above expression that patterns with higher than average fitness tend to increase exponentially over cycles of evolution.

Next, the crossover operation described in the previous section was performed on the selected population with the defined probability of crossover p_c (~ 0.6 to 0.8), to yield the new designs. Finally, a low mutation probability p_m of .009 was used during mutation transformation. In this process, a certain position is chosen along

the string, and based on p_m, the value at that position was simply inverted. The crossover and mutation operations can be disruptive in the evolution process. As shown by Holland [32], in terms of the probabilities of p_c and p_m, the growth or decay in a succeeding generation of a pattern H can be written as follows:

$$m(H,t+1) \geq m(H,t)*\frac{f(H)}{f_{avg}}*(1-p_c*\frac{\delta(H)}{L-1} - O(H)*p_m) \qquad (5\text{-}13)$$

The above expression modifies the growth in population due to reproduction, by considering the effects of crossover and mutation transformations. Equation (5-13) shows that the defining length (δ) plays a significant role in the occurrence of the schema in a succeeding generation. If δ is large for schema that has a relatively low fitness, then its number in succeeding generation would decrease rapidly.

The topology at the end of twenty five generations of evolutions are shown in figure 5-4, and the optimum value of the design variables are given in table 5-1. Figure 5-5 gives the variation of the maximum fitness value in each generation of the genetic search process. As can be seen, there is a general trend towards increasing magnitude for the fitness function.

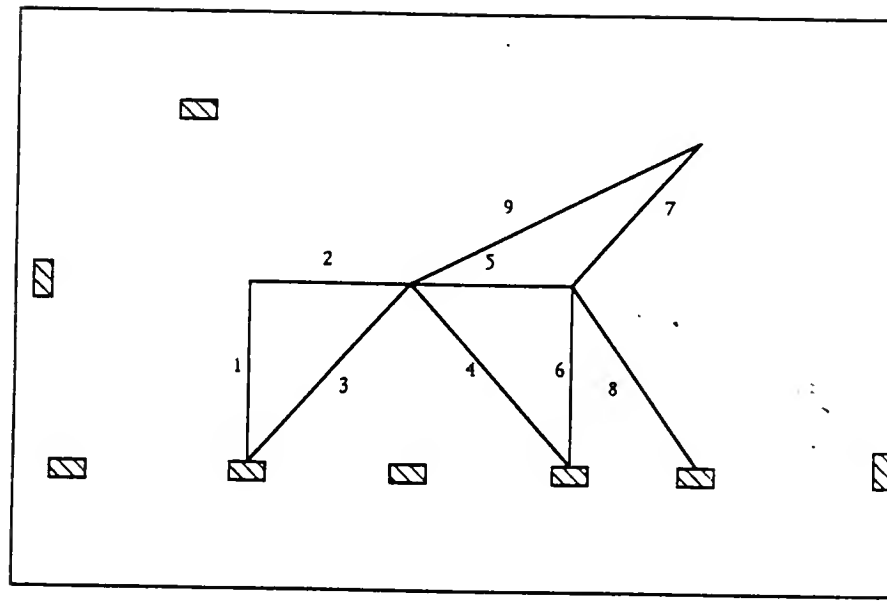


Figure 5-4. Final topology for the example problem.

Table 5-1. Optimized areas for topology in figure 5-4.

Element #	Area (sq. in.)
1	1.6
2	1.9
3	2.8
4	0.9
5	2.5
6	2.2
7	1.6
8	1.1
9	0.9

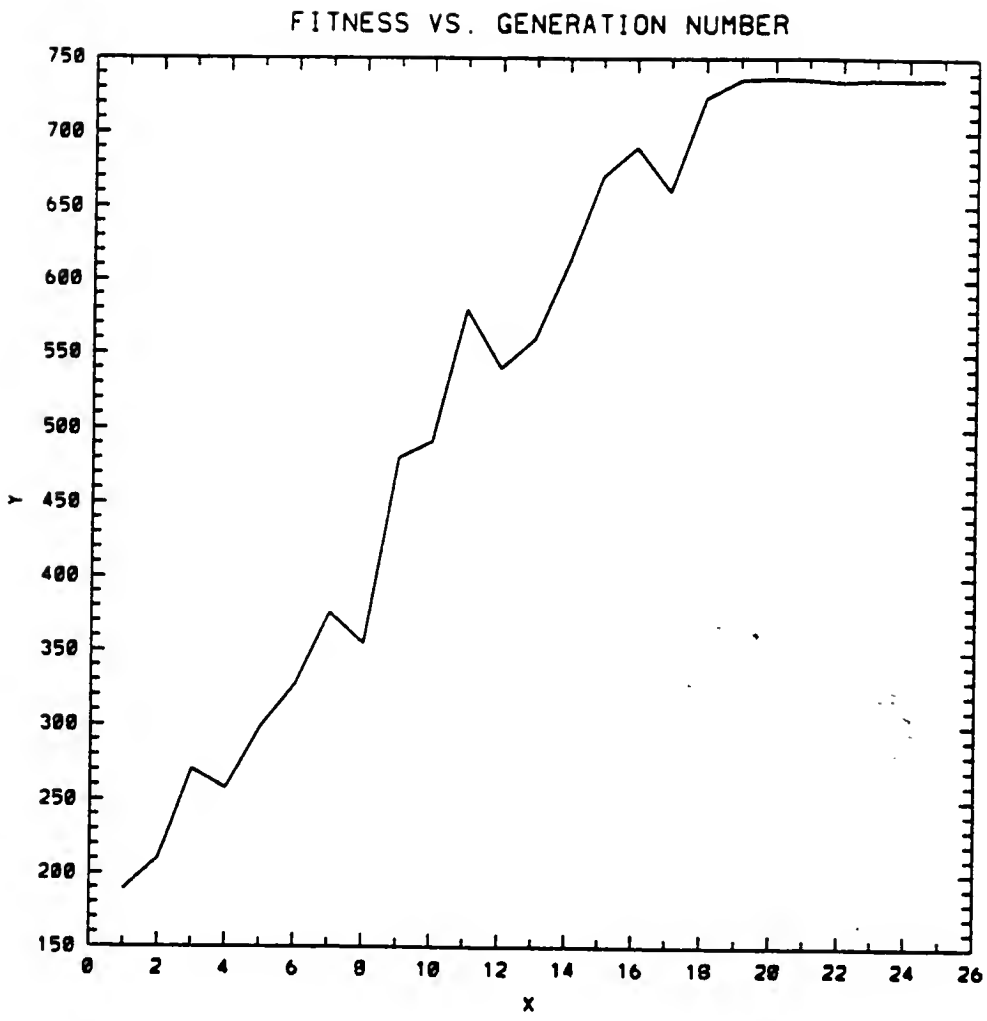


Figure 5-5. Variation of maximum fitness during evolution.

CHAPTER 6
GENETIC AlGORITHMS IN TOPOLOGY GENERATION

The genetic search approach described in the previous chapter was used in the optimal development of structural topology in conjunction with the DP approach. This approach derives its strength principally from the fact that genetic search accommodates a discrete or discontinuous design space in which structural members are both introduced and removed during the design process. Such an adaptation allows a multiple exploration of the topology design space until a satisfactory topology is identified within the guidelines of the design requirements. Subsequent sections of this chapter describe the adaptation of the search in structural topology generation and related implementation details.

Adaptation in Structural Synthesis

The application of genetic algorithms in the generation of structural topology was largely pursued due to the unique representation scheme allowed by the method. Since genetic algorithms work on a representation of the design variables as opposed to the variables themselves, a representation of the presence or absence of a variable can be included along with the dimensions of the variables. If one thinks of a binary string representing the magnitude of a cross sectional area, an additional digit may

be added to the string to indicate the presence of a member. The 0's and 1's can represent the presence or the absence of a member between a load and a support, and hence successive generations will have different topologies created and destroyed until the most fit topology is derived. The central theme of the topology synthesis is that appearance and removal of members is facilitated by the genetic search, provided there is a reliable fitness measuring scheme. In this adaptation, the candidate topologies were derived on the basis of DP method described in the previous chapter, from which a population of least weight topologies is constituted. Once such a population of candidate designs is assembled, genetic search may be used to evolve a generation of improved designs. Ultimately, the least weight topology is identified and prepared for member resizing. The key to a successful implementation of genetic search for topology synthesis is, however, the definition of the fitness function.

As far as implementation issues are concerned, the initial population was seeded with topologies from the SG approach, followed by the formulation of a reliable fitness measuring scheme to take into account all the design requirements coupled with member presence/absence from the topologies.

Integration with SG Approach. Initial population for the genetic search is derived from application of SG based problem decomposition to the structural synthesis problem, as described in the previous chapter. Each individual load is considered a stage, and the different ways of supporting each load constitute the

possible states for each stage. The stabilization of the load in any stage is based on heuristics described in chapter 2. Only optimal weights corresponding to each state in a stage, are propagated to the succeeding stage. Therefore, at the end of the nth stage, where the last load has been stabilized, the output will consist of weights of topologies which are optimal until the n-1-th stage, accrued with the weights corresponding to all the possible ways of stabilizing the last load. Such an array of weights can be rank ordered, and the lowest twenty weights were extracted from the set. Corresponding to each of these weights, the topology is identified by back-propagation, proceeding from the nth stage, and identifying the states that contributed to this topology from the upstream stages. All these topologies were considered as candidate or seed designs for the initial population of the genetic search.

The choice of such candidate designs for the initial population has a direct bearing on the number of evaluations necessary. Powell et al. [33] report on the influence of initial seeding on the number of evaluations required during genetic search. Population seeding is used to overcome the efficiency problem with genetic search. If possible candidate designs (preferred) are known, then randomizing the initial population which leads to a greater number of evaluations can be avoided.

Fitness Function Formulation. The task of formulating a fitness function is perhaps the most important component in the topology generation process using genetic search. Since all the evolution processes are in one way or other dependent

on the fitness function, a careful consideration must be given to this formulation. The fitness function has to take into account all the factors that go into the determination of relative merits of the different members in the population. However, an option exists of not including some of the more complex design constraints in the fitness function, as they can also be handled at the optimization level.

In the present topology generation problem, the fitness function had to meet three requirements. The first was to achieve a least weight topology for the given design domain, using nominal cross section members. Second, the topologies had to be verified for satisfaction of design constraints. The third and the most important, was the ability to accomodate the appearance and disappearance of members in different topologies. With these requirements in mind, the objective was formulated as follows:

$$z = w + R * \sum \langle g_i \rangle^2 + S * \sum \frac{u}{v} \qquad (6\text{-}1)$$

where, 'w' is the weight of the structure; 'R' a penalty parameter associated with the violated design constraints; 'g_i' is the design constraint; 'S' is a penalty parameter with a value set to 100, and 'u' and 'v' are integers which will be explained in the following discussions. The above objective function was transformed into a fitness function z^* which can be maximized during the genetic search (eqn. 5-8).

The representation of structural members which are introduced or removed

had to be transformed into a quantitative form, so as to include this process in the fitness function. This was accomplished in conjunction with stabilizing each load in the design problem. Each load was stabilized by introducing members to any of the 'n ' supports that were available. Also available for each load were the previously stabilized loads, which were now considered as simple supports.

If there are 'm' loads and 'n' supports in the design domain, then for the first load to be stabilized, there are 'n' members that can be introduced. For the second load that is considered, there are 'n+1' members that can be introduced, where an additional member can be introduced between the load and the previously stabilized load point. Therefore, for the mth load, the number of possible members that can be introduced is n+m-1. For each load, a string of length corresponding to the number of possible members that can support this load, is chosen. A '0' denotes a member absence and '1', its presence. The bit string representation for the topology is a concatenation of such strings for all the loads. Therefore, the total bit string length to represent a topology, for a 'm' load and 'n' support domain is given by

$$n + (n+1) + (n+2) + \ldots (n+m-1) = m\ (n + (m-1)/2) \qquad (6\text{-}3)$$

For each load, the bit string represents an increasing order of lengths (weights) of members when moving from left position to right in the string. The cumulative of the position numbers containing a '1' in the bit string is the variable 'u' in equation 6-1. The number of digits in a string is the variable 'v' in equation 6-5. An illustration is provided below.

string : 1001001001

'u' : 1+4+7+10 = 22

'v' : 10

The successful formulation of the fitness function leads to the implementation of the design process which is explained in the next section.

Implementation and Examples

The initial population of topologies was derived from a DP approach as described previously. For each of the topologies, nominal cross sectional area members were introduced for analysis. A crossover rate of 0.65 and a mutation rate of .009 were assumed for the present simulation. An evolution of twenty generations was conducted in this topology generation exercise.

The example problem considered is the same as in chapter 5, and is shown in figure 6-1. Figures 6-2, 6-3 and 6-4 depict topologies at the end of first, tenth and twentieth generations. All the topologies in any generated population were checked for stability considerations. This was accomplished by the examination of each topology in the population using the previously developed knowledge-base. The stabilization of each load was checked and member(s) introduced where necessary to ensure stability.

The least weight topology is highlighted in the figure. Optimization was performed to resize the member cross sectional areas using genetic search for twenty

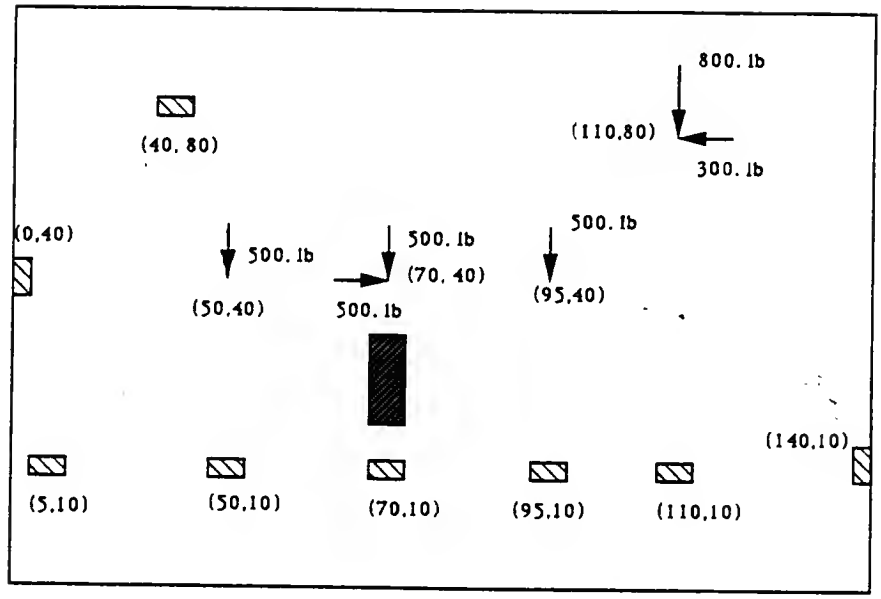


Figure 6-1. Example design domain.

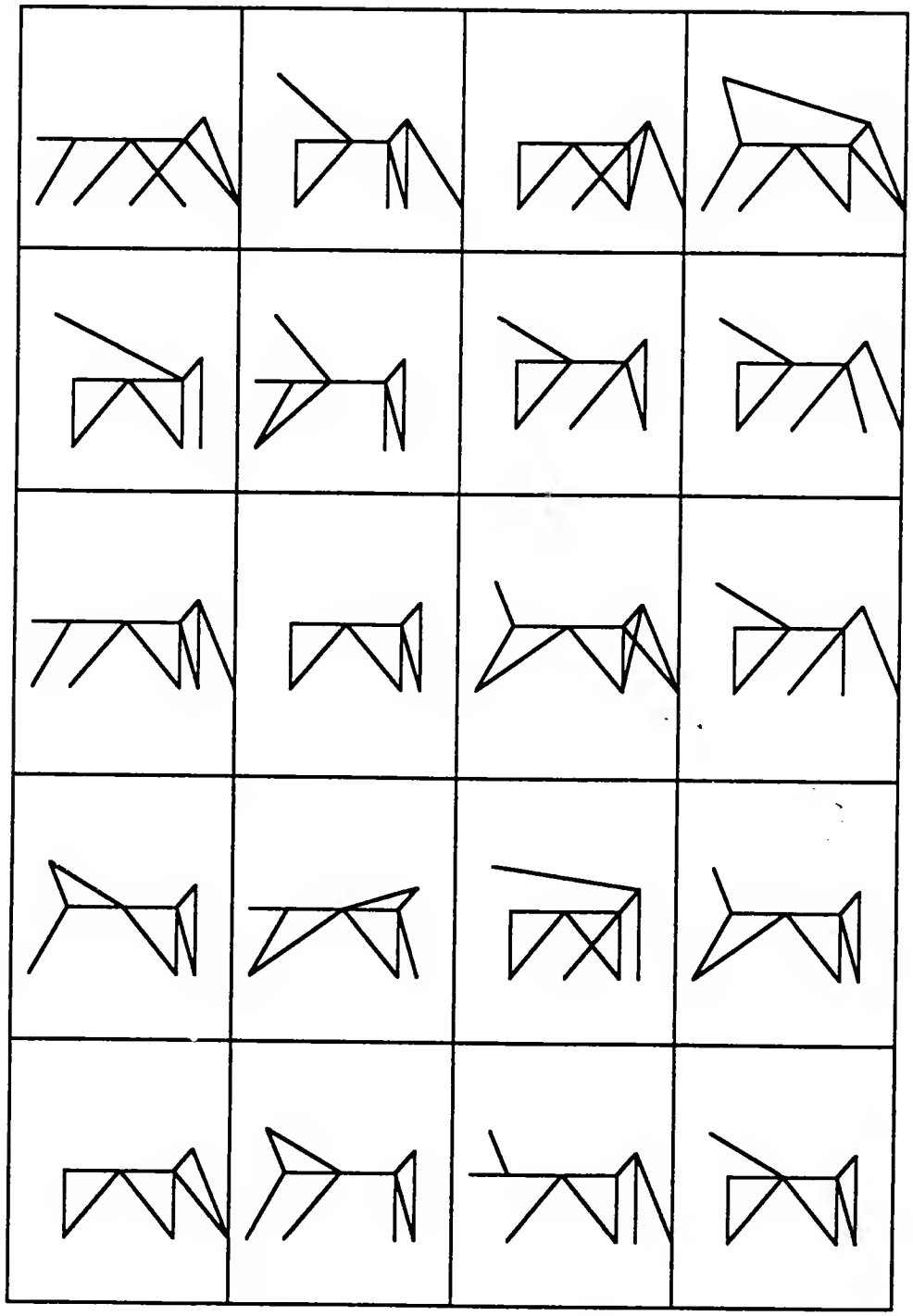


Figure 6-2. Topologies in generation #1.

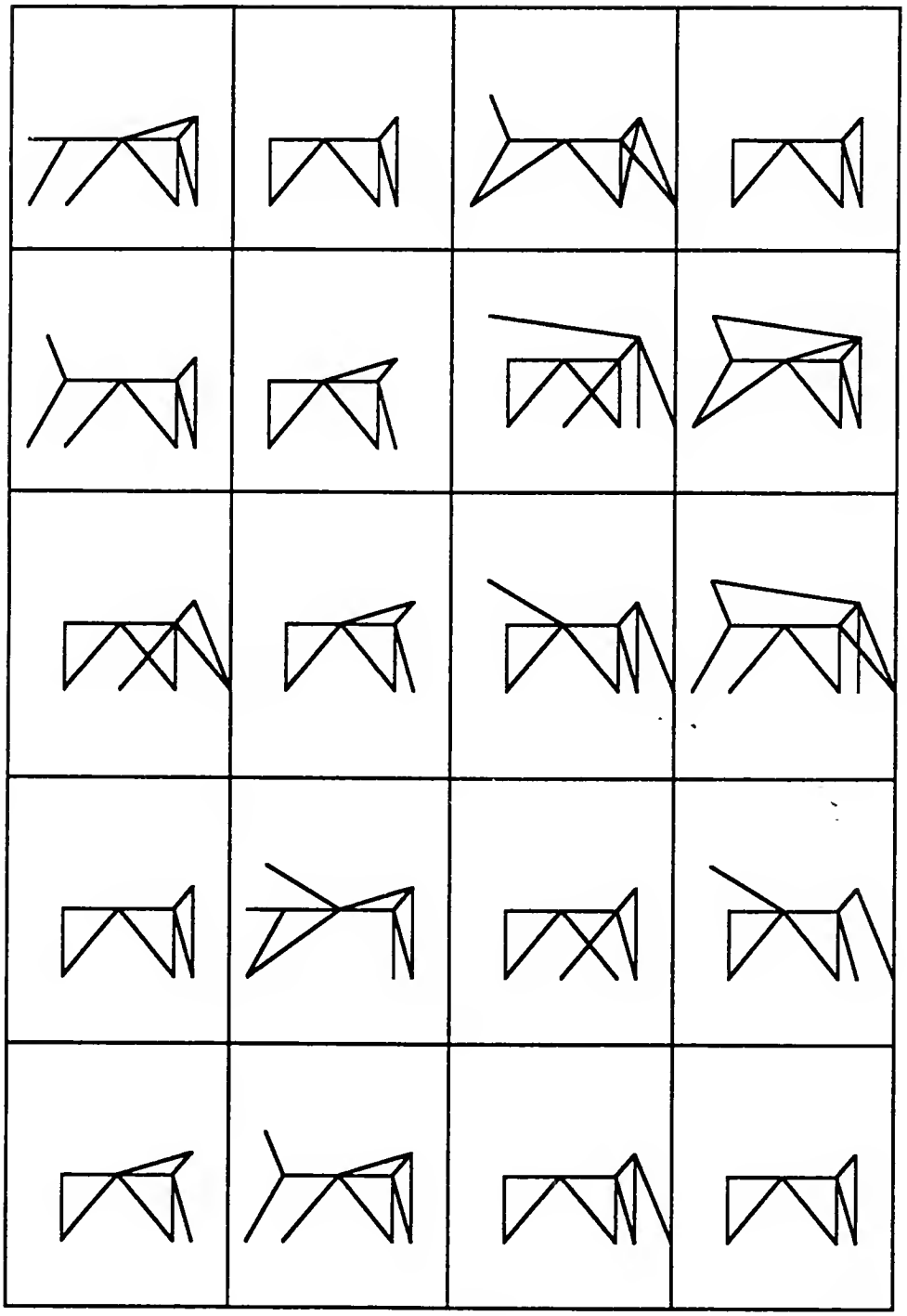


Figure 6-3. Topologies in generation #10.

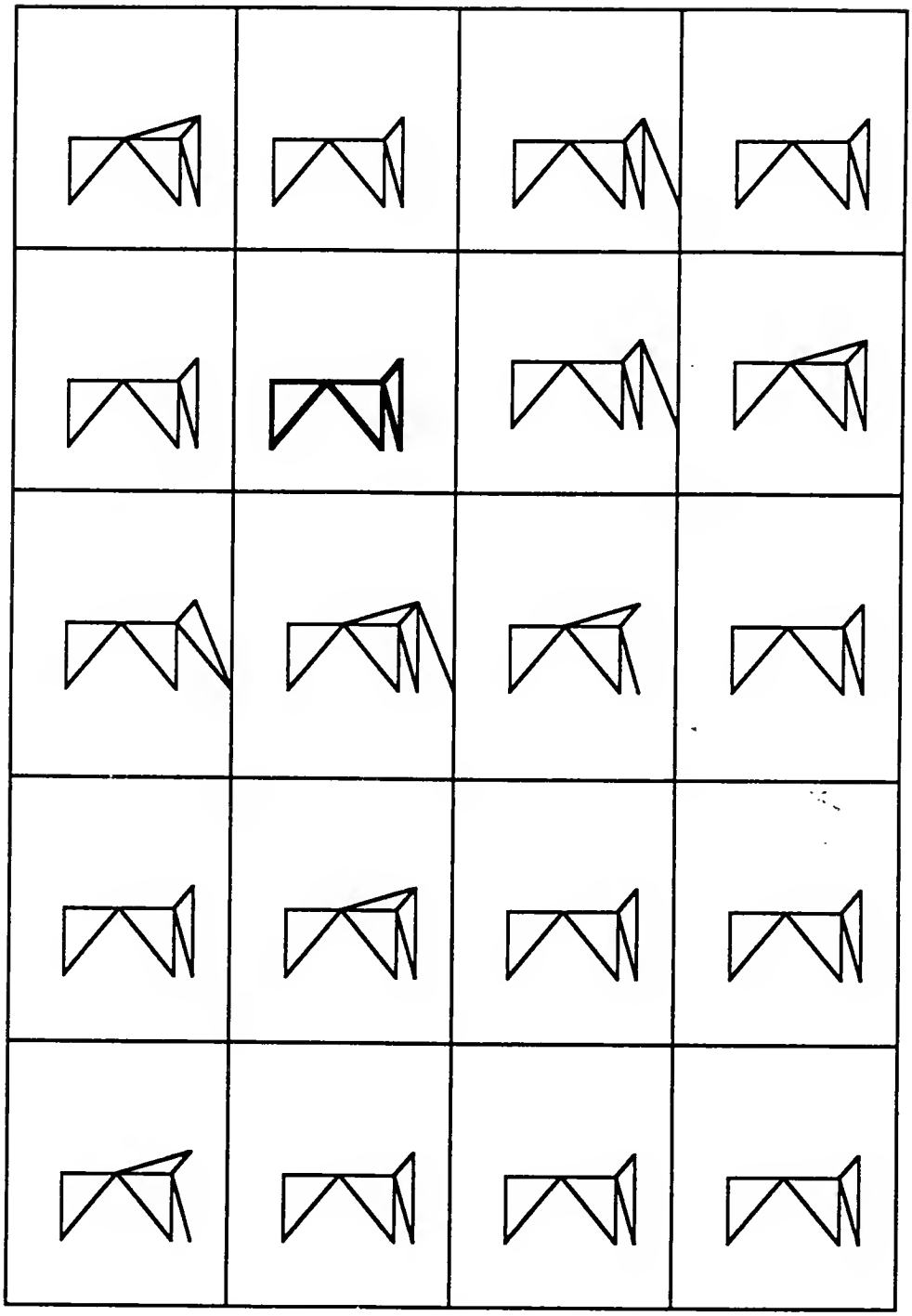


Figure 6-4. Topologies in generation #20.

generations, with a crossover rate of 0.65 and a mutation rate of 0.009. The final design is shown in figure 6-5. Table 6-1 contains the optimized cross sectional areas of members in figure 6-5.

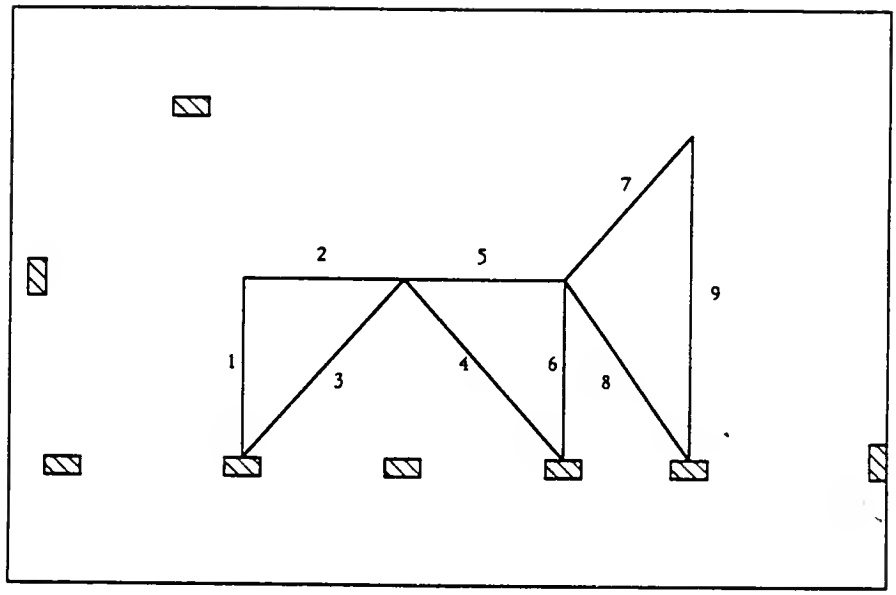


Figure 6-5. Final topology.

Table 6-1. Optimized areas for example in figure 6-5.

Element #	Area (sq. in.)
1	1.4
2	2.0
3	2.8
4	0.9
5	2.3
6	2.2
7	0.6
8	0.3
9	1.0

CHAPTER 7
CLOSING REMARKS

The successful development and implementation of the design methodology has brought into focus a variety of observations. In this chapter, some of the observations that became evident during the course of research, and their implications on future research are enumerated. Concluding observations include a careful scrutiny of the architecture, the decomposition methods, and experiences with the implementation, including both advantages and drawbacks.

Conclusions

The architecture proposed for the design procedure was successfully implemented for the development of 'near-optimal' load bearing structures. The design architecture provided for a convenient linking of algorithmic procedures and heuristics, within the framework of an automated design system. The most important feature of the system is a considerably large knowledge base that has been incorporated to overcome the drawbacks of a consultative system. In comparison with a typical consultative system, the interactive feature was deemphasized with the intent to improve productivity in case of real world designs. At the same time, however, it is important to indicate that this automation is only built in to remove

the necessity of having a user respond to questions for which the response may be obvious. Instead, the system uses algorithmic analysis to arrive at a response to its query. To this extent, lack of interactiveness should not be viewed as an attempt to seize initiative from the designer.

The system clearly identifies the various levels at which information must be organized in a typical design process. The implicit recognition of the three levels formalizes the organization of the various tools and validation procedures, depending on their functionalities and hierarchy of execution. The organization can be tailored to most design applications, with changes occurring in the tools for analysis and knowledge bases. The domain independence associated with the framework can be attributed to the flexibility in interactions between different levels and between modules. The interactions between the knowledge level and the function level were such that unless an effort is made to identify them as separate entities, they were interspread appropriately in the implementation to achieve uniformity. In other words, the decision making capabilities provided by the knowledge bases were treated just like any other procedure in the design system.

The preliminary design procedure which is crucial in a design process, was represented by a large knowledge base. The resulting abstraction can be analyzed both quantitatively and qualitatively. The quantitative approach was adopted in this analysis, and provided a rationale for the acceptance or the rejection of a design.

The preliminary design procedure generated a taxonomy of design abstractions. A tree-like deduction approach identified the best design among the

taxonomy of designs, based on satisfaction of design requirements. The tree-like representation of various designs automatically provided for hierarchical classification and a convenient method for constraint propagation. The implementation of the tree search helped in maintaining a history of designs, each with its own merits.

The adhoc decomposition procedure used during the initial phases of the implementation provided satisfactory structural topologies. The intermediate designs obtained during the generation process varied as the order of constraint satisfaction was changed, and this can be attributed to the fact that at every instant a constraint stands violated, the search moved to different branch in the tree, yielding a different design. But, since the outcome was always aimed at achieving a least weight topology prior to optimization, all the different orders of constraint satisfaction yielded the same topology which is least in weight so far, and satisfied all the design requirements. This is due to the fact that for a given design domain, there is only one least weight topology which satisfies all the constraints. Note that when multiple feasible solutions are available, the order in which constraints are satisfied will influence the final outcome.

The heuristics directed optimization approach provided satisfactory results. The optimum weights that were realized were greater than the true optimum by about 15 % - 20%, but with a significant decrease in the number of function evaluations. The heuristics based optimization is a viable approach when dealing with large optimization problems, and in cases where only near optimal estimates of the objective function are desired.

The application of global sensitivity analysis as an approach to problem decomposition was found to be effective in handling structural synthesis problems, wherein inherent coupling between subproblems was retained throughout the analysis. In the domain of structural design, it reduced the complexity of analysis of a large structural problem by posing the larger problem as many smaller simpler problems. Furthermore, heuristics directed integration of the partial structures were based on the results of the decomposition, which provided insight into weight and constraint variations. The only drawback that can be attributed to this method was the large volume of computation required during analysis.

The second approach to problem decomposition during structural synthesis problems was based on dynamic programming approach. The advantage was an efficient management of the size of the design space. Since, only the stagewise optimal states were considered for further analysis, this procedure helped significantly prune the design alternatives to manageable levels. In the design example considered for this approach, the total number of possible topologies was in excess of fifty. However, when using dynamic programming this was reduced to less than twenty, which included the least weight topology. This approach can be very productive in case of three dimensional structural synthesis problems, where there is a possibility of combinatorial explosion.

A different approach to structural synthesis on the basis of genetic algorithms identified the least weight topology in an efficient manner. The behavior of the fitness function was typical in comparison to other genetic search problems. The

formulation of the fitness function was the key to this approach. Among other things, it took into consideration the variation in topologies in a given population by providing penalties for members according to their respective length. The performance was further improved by providing the initial population based on a dynamic programming procedure, which introduced good initial designs in the population. The combination of dynamic programming and genetic search process proved to be a very effective structural synthesis procedure, and holds promise for further improvement in the efficiency of the design process.

The decomposition methods have been implemented successfully and the results are promising, providing hope for further investigation into this approach in case of large structural problems. The decomposition methods coupled with heuristics form an efficient tool for structural synthesis.

Recommendations for Further Research

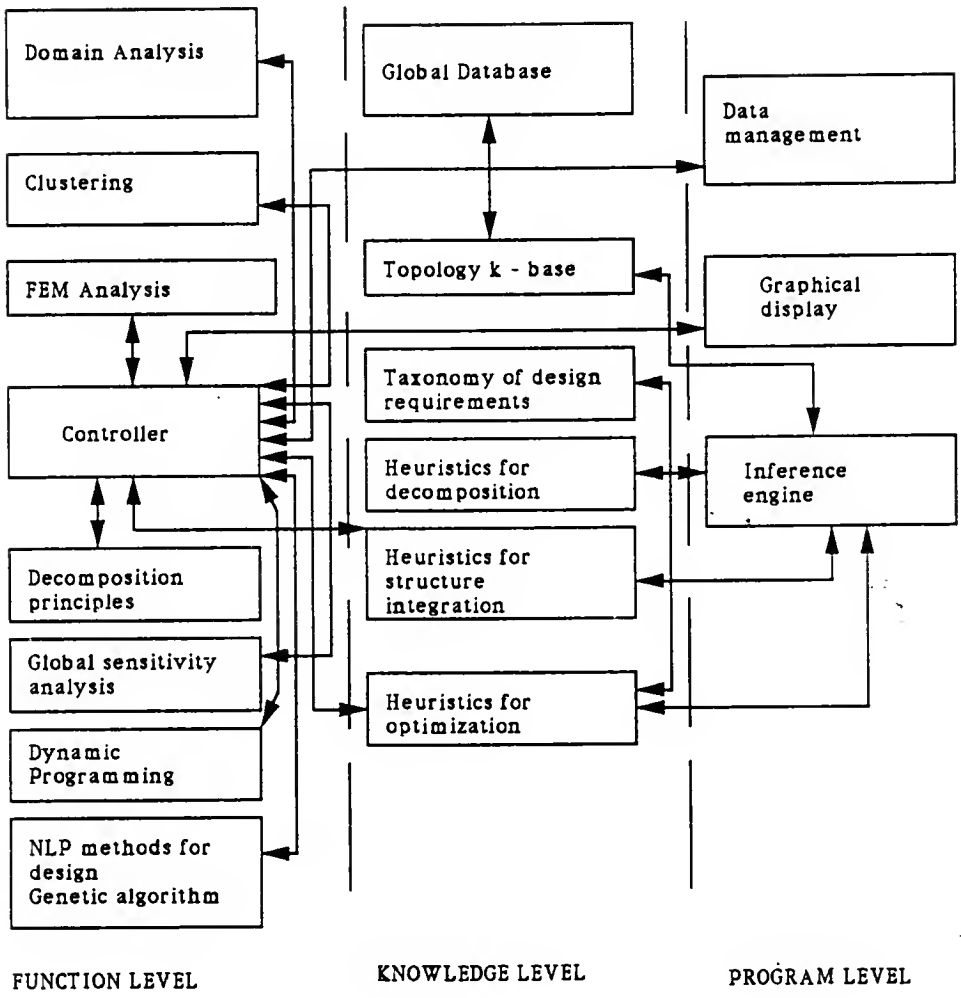
The design methods for structural synthesis have proven to be computationally demanding. Therefore, it is imperative to look for methods which can reduce the computation, at the same time maintain an acceptable level of abstraction capabilities without sacrificing the theoretical implications. One such approach can be the use of object oriented approach to structural synthesis. This approach can be used to efficiently portray the characteristics of each member, domain features, and last but not the least, their connectivity information. Since the use of a knowledge

based approach to structural synthesis has been proved viable, use of object oriented approach can make it even more powerful.

Decomposition methods have been successfully tailored for structural synthesis problems in this work, while limiting the domain to two dimensions. These approaches were enhanced by the extensive application of heuristics. The approaches have to be explored for realistic three dimensional structural synthesis problems. The problem of combinatorial explosions in case of this increased complexity, has to be addressed from the point of design space management. In the work reported here, dynamic programming methods were successfully employed for this purpose, and hold promise for extension to a 3-dimensional space. Global sensitivity based decomposition was found to yield very strong quantitative basis for integration of partial structures. Here again, the problem is one of extensive computation that goes with the method. Alternative approximation methods have to be explored to reduce the analysis time, if this method needs to be given preference over the dynamic programming based approach. The applicability of this method to develop three dimensional structural systems has to be investigated.

The adaptability of genetic algorithms in a structural synthesis process has been demonstrated in this work. The same approach can be extended to the design of large structures. The newly developed string representations have to be investigated from an applicability viewpoint, for structural topology generation. Also, a simultaneous topology synthesis and optimization can be investigated in the context of the genetic search approach.

APPENDIX A
ORGANIZATION OF THE DESIGN SYSTEM



APPENDIX B
TYPICAL RULE IN THE KNOWLEDGE BASE

Typical rules in the knowledge base for topology generation are shown. A brief comment is provided following each statement for the first rule.

```
(defrule rule6                                          $name of rule
?k <-(new-id ?g $?a                                     $variable g,array a,
                                                        statement k
      (?num)                                            $variable num, this is
                                                        the load#
=>
      (bind ?a (nth 7 $?))                              $7th value in array a is
                                                        variable a, nearest support#
      (bind ?b (nth 8 $?a))                             $8th value in array a is
                                                        variable b,support type
      (bind ?c (nth 14 $?a))                            $second nearest support#
      (bind ?d (nth 15 $?a))                            $support type
      (if (= ?b 1)                                      $if first support is clamp
        then
            (bind ?f 1)                                 $member # 1
            (format topo "%4d %4d %4d b %n" ?f ?num ?a)
                                                        $writing the member#,
                                                        load#, support# and
                                                        beam type element to an
                                                        output file
      else
            (if (= ?d 1)                                $check if second
                                                        support is clamp
          then
                (format topo "%4d %4d %4d b %n" ?f ?num ?c)
                                                        $write the member#,
                                                        load#, support# and
                                                        beam type to an output file
            else
                (format topo "%4d %4d %4d b %n" ?f ?num ?a)
                (format topo "%4d %4d %4d b %n" ?f ?num ?c)))
```

$write the member#, load#, support# and beam type elements to connect the load to both the supports, to an output file.

APPENDIX C
EXECUTION OF DESIGN SYSTEM

A detailed description of the execution process of the design process is documented below.

$create an input data file. This file contains information in the following order.

# of loads
# of supports
# of distributed loads
# of forbidden areas
coordinates of loads
direction cosines of loads
magnitudes of loads
coordinates and type of supports
coordinates of the end nodes and magnitudes of distributed loads
coordinates of the vertices of all forbidden areas

The constraints are given in a separate file.

$run the design domain evaluation module. This creates an extended database and a facts file of the form shown below.

(deffacts fact1
(new-id gen1 load#, load location, x component of load, y component of load, moment, support#, type, angle, distance, ...))

$execute the topology knowledge base in the CLIPS environment. The procedure includes executing CLIPS, loading facts and rules files, and executing the rules. The output for exampleshown in figure 2 is given below

1	1	4	b
2	1	2	b to nearest load
3	2	5	t
4	2	3	b to nearest load
5	3	6	b

The columns refer to member #, load #, support #/load #, member type respectively.

$using the above output file, an EAL runstream is created to perform an analysis of the structure. The displacement values, first fundamental frequency and the stress values are extracted from the analysis.

$the displacement values are compared to the constraint allowables. If the constraint is satisfied, then frequency constraint is checked. Otherwise the topology is refined heuristically.

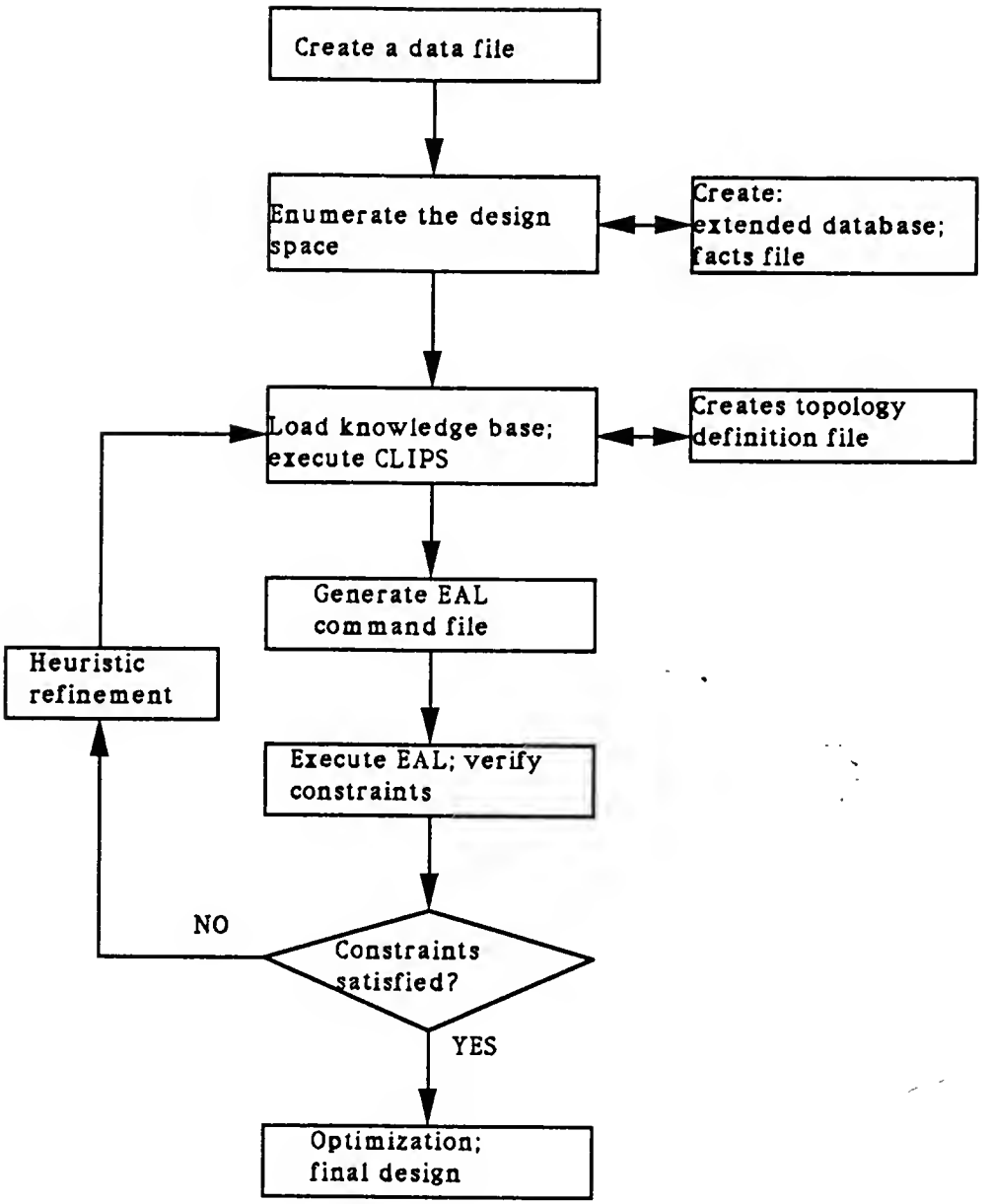
$an EAL runstream is created and the structure analyzed.

$the frequency values are verified for constraint satisfaction. If violated, then the topology is refined heuristically until the constraint is satisfied.

$the stress constraints are verified. If any violation is present then topology is modified by heuristics.

$the successful structure is analyzed using EAL. Each design variable is perturbed by 3% of its design value and the sensitivities of the constraints and the objective with respect to the design variable is evaluated. Design is updated on the basis of sensitivities to a feasible point. The sensitivities are recomputed.

$heuristic optimization takes place in CLIPS environment. CLIPS is executed followed by the loading and execution of the rules file. The output file format is a 2 X n array, with the design variables in the first column and its numerical value in the second column.


Create a data file
Enumerate the design space
Create: extended database; facts file
Load knowledge base; execute CLIPS
Creates topology definition file
Generate EAL command file
Heuristic refinement
Execute EAL; verify constraints
NO
Constraints satisfied?
YES
Optimization; final design

REFERENCES

1. Ohsuga, S., "Conceptual Design of CAD Systems Involving Knowledge Bases." Knowledge Engineering in Computer Aided Design, Ed. J. S. Gero, North-Holland, Amsterdam, 1985.

2. Airchinnigh, M., "CAD, KE and Ada." Knowledge Engineering in Computer Aided Design, Ed. J. S. Gero, North-Holland, Amsterdam, 1985.

3. Rehak, D.R., Howard, H.C., and Sriram, D., "Architecture for an Integrated Knowledge Based Environment for Structural Engineering Applications." Knowledge Engineering in Computer Aided Design, Ed. J. S. Gero, North-Holland, Amsterdam, 1985.

4. Shah, J. J., and Pandit, L., "DEZINER - An Expert System for Conceptual Form Design of Structural Parts" Proceedings of the ASME International Computers in Engineering Conference, Vol. 1, Chicago, July, 1986.

5. Nevill, G. E., Jr., Jackson, L. A., and Clinton, J. H., "Automated Hierarchical Planning for Structural Design" Proceedings of the ASME Computers in Engineering Conference, San Francisco, August, 1988.

6. Brown, J. P., "Managing Interacting Design Subproblems in Incremental Preliminary Design." M.S. Thesis, Department of Aerospace Engineering, Mechanics and Engineering Science, University of Florida, August, 1988.

7. Brown, D. C., and Chandrasekharan, B., "Expert Systems for a Class of Mechanical Design Activity." Knowledge Engineering in Computer Aided Design, Ed. by J. S. Gero, North-Holland, Amsterdam, 1985.

8. Davies, B. J., and Darbyshire, I., "EXCAP: An Expert Generative Process Planning System." Knowledge Engineering in Computer Aided Design, Ed. J. S. Gero, North-Holland, Amsterdam, 1985.

9. Rooney, F. M., and Smith, E. S., "Artificial Intelligence in Engineering Design.", Computers and Structures, Vol. 14, pp. 279-288, 1983.

10. Dixon, J. R., and Simmons, M. R., "Computers That Design: Expert Systems for Mechanical Engineers." CIME, pp. 10-18, November, 1983.

11. Liu, S. C-Y., "An Intelligent Framework for Engineering Decision-Making." SAE International Congress and Exposition, Detroit, MI, 1987.

12. Arora, J. S., and Baenzinger, G., "Uses of AI in Design Optimization." AIAA Journal, Vol. 25, pp. 834-846, 1985.

13. Bennett, J. S., and Englemore, R. S., "SACON: A Knowledge Based Consultant for Structural Analysis." Proceedings of the Sixth Joint Conference on Artificial Intelligence, Tokyo, 1979.

14. Rogers, J., and Barthelemy, J., " An Expert System for Choosing the Best Combination of Options in the General Purpose Program for Automated Design Synthesis." NASA Technical Memorandum, 1985.

15. Sriram, D., Maher, M. L., and Fenves, S. J., "Knowledge Based Expert System in Structural Design." Computers and Structures, Vol. 20., pp. 1-9, 1985.

16. Lusth, J. C., Schick, W. R., and Singh, G. D., "An Expert System for Arc Welding." SAE International Congress and Exposition, Detroit, MI, 1987.

17. Liu. S. C-Y., "Application of Knowledge Based Expert System in Automated Manufacturing.", Proceedings of 1985 ASME International Computers in Engineering Conference, Boston, MA, 1985.

18. Shodhan, R., and Talavage, J. J., "A Combined Simulation-Artificial Intelligence Approach for Manufacturing system Design." SAE International Congress and Exposition, Detroit, MI, 1987.

19. Phillips, R. H., Zhou, X-D., and Mouleeswaran, C. B., "An Artificial Intelligence Approach to Intelligent CAD and CAM Through Generative Process Planning." Proceedings of ASME International Computers in Engineering Conference, Chicago, IL, 1986.

20. Murthy, S. K., Navathe, S. B., Cornelio, A., "Organizing Engineering and Design Techniques." Proceedings of the Fifth IEEE International Symposium on Intelligent Control, Philadelphia, PA, 1990.

21. Tong, C., "Towards an Engineering Science of Artificial Intelligence.", Journal of Engineering Science, Vol. 2, No. 3, pp. 133-166, 1987.

22. CLIPS, Reference Manual, JSC-22948, NASA Johnson Space Center, Houstan, TX, April 1988.

23. Whetstone, D., EAL-Reference Manual, NASA CR 145098-1, February 1977.

24. Sobieski, J. E., "Sensitivity of Complex Internally Coupled systems." AIAA Journal, Vol. 28, pp. 153-160, 1990.

25. Hajela, P., "Further Developments in the Controlled Growth Approach for Optimal Structural synthesis." Paper No. 82-DET-62, ASME, New York, NY.

26. Vanderplaats, G., ADS-A FORTRAN Program for Automated Design Synthesis-Ver1.10, Contract Report 177985, NASA, Cleveland, OH, September 1985.

27. Kirsch, U., Optimum Structural Design-Concepts, Methods and Applications. McGraw-Hill, New York, NY, 1981.

28. Bellman, J. R., Dynamic Programming. Princeton University Press, Princeton, NJ, 1957.

29. Hajela, P., "Genetic Search - An Approach to the Nonconvex Optimization Problem." to appear in AIAA Journal 1991.

30. Hajela, P., "Genetic algorithms in Automated Structural Synthesis." Vol II, NATO Advanced Study Institute on Optimization and Decision Support Systems, Kluwer Academic, Boston, MA, 1989.

31. Bethke, A. D., "Genetic Algorithms as Function Optimizers." Ph.D Dissertation, University of Michigan, Ann Arbor, MI, 1980.

32. Holland, J. R., Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI, 1975.

33. Powell, D. J., Tong, S. S., and Skolnick, M. M., "EnGENEous-Domain Independent Machine Learning for Design Optimization." Proceedings of the Third International Conference on Genetic Algorithms, Washington DC, 1989.

BIOGRAPHICAL SKETCH

The author graduated with a B.S (honors) degree in mechanical engineering from the University of Madras, India in 1984. He received M.S. degree in Mechanical Engineering from Southern Illinois University at Carbondale in 1986. His research interests are design automation, knowledge based systems, optimization, and genetic algorithms.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Prabhat Hajela, Chair
Associate Professor of Aerospace Engineering, Mechanics and Engineering Science

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Gale E. Nevill, Jr.
Professor of Aerospace Engineering, Mechanics and Engineering Science

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

C. T. Sun
Professor of Aerospace Engineering, Mechanics and Engineering Science

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

David C. Zimmerman
Assistant Professor of Aerospace Engineering, Mechanics and Engineering Science

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Shamkant Navathe
Professor of Computer and Information Sciences

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1991

for Winfred M. Phillips
Dean, College of Engineering

Madelyn M. Lockhart
Dean, Graduate School